UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

Utilizzo e sperimentazione di una rete neurale convoluzionale per analizzare le serie temporali maggiormente anticorrelate dell'elettroencefalogramma e riconoscere l'attività cardiaca

Relatori:

Ing. Antonio Luca Alfeo

Prof. Mario G.C.A. Cimino

Candidato:

Tommaso Falaschi

ANNO ACCADEMICO 2023/2024

Abstract

Lo studio proposto pone l'attenzione sull'utilizzo dell'*Intelligenza Artificiale* per l'analisi e il riconoscimento dei segnali fisiologici, processo che permette l'identificazione di eventuali pattern e anomalie nei dati raccolti dai segnali biologici delle persone. L'applicazione principale può essere trovata in campo medico per realizzare diagnosi, per il monitoraggio della salute di specifici soggetti o per lo studio e la comprensione dei processi fisiologici del corpo umano.
I dati raccolti per questo lavoro fanno parte di uno studio realizzato dall'*Università di Pisa* e dal *Politecnico di Milano* per l'analisi dell'interazione Cervello-Cuore. I dati sono relativi a 26 soggetti sani, di cui sono stati raccolti EEG, attività respiratoria ed ECG.
L'obbiettivo dello studio è stato quello di trovare un modello capace di riconoscere, con la massima accuratezza possibile, la presenza *(peak)* o l'assenza *(non-peak)* di picchi nei battiti cardiaci dei dati raccolti. Lo scopo finale era quello di riconoscere la *Heart Rate Variability (HAR)* dalle EEG.
Il problema trattato è stato dunque quello di una classificazione binaria con lo scopo di massimizzare l'accuratezza del modello nel riconoscere i dati. Questa metrica non è ovviamente dipendente unicamente dal tipo di modello utilizzato, di cui ne possono essere utilizzati molti e ciascuno con un funzionamento differente, ma anche dalle parametrizzazioni utilizzate all'interno del modello.
Lo studio si è dunque prima focalizzato sull'analisi e la ricerca del modello per il trattamento dei dati, e successivamente sullala fase di test delle prestazioni del modello al variare di alcune parametrizzazioni, scelte in base alla loro utilità e influenza nel modello, che potessero migliorare l'accuratezza dello stesso.
I risultati ottenuti sono stati utili per capire se un modello potesse essere utilizzato per questi dataset, e quali parametrizzazioni possono portare ad un incremento delle prestazioni e quali invece ad un decremento. In questo caso le prestazioni migliori sono state ottenute con l'utilizzo di una *Convolutional Neural Network* dopo l'esplorazione di vari iperparametri da utilizzare.

Indice

1 Introduzione **5**

2 Related Works **7**
2.1 Utilizzo di modelli di Machine Learning per l'analisi di segnali fisiologici 7
2.1.1 Applicazioni per il riconoscimento dell' HRV da EEG 8
2.2 Ottimizzazione degli iperparametri per un modello 9

3 Design e Implementazione **11**
3.1 Design 11
3.1.1 Classificazione 11
3.1.2 Modelli utilizzati 11
3.1.3 Pre-processing dei dati 12
3.1.4 Ottimizzazione degli iperparametri 12
3.1.5 Metriche per l'analisi prestazionale del modello 14
3.2 Implementazione 15
3.2.1 Use case 16

4 Case Study **18**
4.1 Dataset 18
4.1.1 Descrizione del dataset 18
4.1.2 Processing dei dati EEG 19
4.1.3 Processing dei dati ECG 20
4.2 Librerie utilizzate 21

5 Risultati Sperimentali **23**
5.1 Ottimizzazione di una Convolutional Neural Network Unidimensionale 23
5.1.1 Ottimizzazione dell'iperparametro batch_size 23
5.1.2 Ottimizzazione di Kernel_Size, Pool_Size e ottimizzatore . . 25
5.1.3 Valutazione delle prestazioni al variare del metodo di pooling . 28
5.1.4 Ottimizzazione di filters, epochs, learning_rate e momentum . 29
5.2 Utilizzo di una Convolutional Neural Network Bidimensionale 34
5.3 Utilizzo del modello Learning Shapelets 35
5.4 Prestazioni di una CNN1D con una diversa estrazione dei dati 37

6 Conclusioni **39**
6.1 Considerazioni finali 39
6.2 Applicazioni e sviluppi futuri 41

Bibliografia **42**

Capitolo 1

Introduzione

Grazie al notevole sviluppo dell'*Intelligenza Artificiale* negli ultimi anni è stato possibile applicarla ad un numero sempre crescente di settori. Tra questi possiamo trovare ad esempio il campo medico per l'assistenza nelle diagnosi, il monitoraggio dei pazienti con particolari malattie o disturbi, nel settore dell'industria dove l'IA può essere utilizzata per effettuare il controllo della qualità dei prodotti, nel settore automobilistico grazie allo sviluppo di veicoli a guida autonoma e altri sistemi che possono garantire una maggiore sicurezza dei guidatori, e in molti altri settori molto differenti, all'interno dei quali l'IA può essere utilizzata per migliorare notevolmente la qualità del servizio.
In questo contesto la ricerca si è concentrata sul settore medico, più specificatamente sull'analisi dei segnali fisiologici per la comprensione dei processi biologici umani. In particolare i segnali presi in considerazione fanno riferimento allo studio dell'interazione tra cervello e cuore *(Brain Heart Interaction - BHI)* sotto stress termico. L'utilizzo dell'*Intelligenza Artificiale* in questo campo consente di identificare e interpretare segnali che possono essere anche di notevole complessità, difficilmente interpretabili da un essere umano, migliorando così la capacità di individuare possibili pattern rilevanti all'interno dei dati misurati. Inoltre con l'utilizzo di un modello ad alte prestazioni è possibile automattizare il processo di analisi dei dati raccolti, permettendo così di risparmiare tempo e risorse in un'operazione che sarebbe altrimenti dispendiosa e potrebbe essere soggetta a maggiori errori ottendo una minore accuratezza nei risultati.
Nel caso di studio considerato risultano essere presenti queste problematiche appena descritte, dato il riconoscimento di HRV da EEG prendono in considerazione dati rilevati da due fonti diverse, il cuore per HRV e il cervello per l'EEG. Questo porta dunque a non poter fare una analisi diretta delle misurazioni effettuate, ma a dover studiare i segnali della BHI. Grazie all'integrazione dell'*Intelligenza Artificiale* in questo settore è possibile realizzare modelli che permettono di implementare importanti funzionalità come la predizione di malattie debilitanti come quelle cardiovascolari o l'epilessia oppure il riconoscimento delle emozioni. Per l'analisi dei segnali fisiologici sono molteplici i possibili modelli da utilizzare come possono esse-

re *Random Forest*, *MLP Classifier* o *Support Vector Machines*. In questo specifico caso i modelli utilizzati sono stati principalmente 2, ovvero la *Convolutional Neural Network (CNN)*, uno specifico modello di *Artificial Neural Network (ANN)*, basato sull'utilizzo di neuroni e dell'operazione di convoluzione, e il modello Learning Shapelets più specifico per la classificazione di timeseries tramite l'identificazione di forme particolari. Sono state utilizzate due tipi di CNN, sia monodimensionale *(CNN1D)* che bidimensionale *(CNN2D)*.
Con l'utilizzo dell'*Intelligenza Artificiale* nel campo medico abbiamo bisogno che i modelli siano affidabili con prestazioni sempre elevate, dato che se l'accuratezza del modello non dovesse essere sufficiente le conseguenze che ne potrebbero scaturire sarebbero gravi, come potrebbe essere l'errata diagnosi di una malattia ad un paziente.
Il problema preso in considerazione è stato quello della classificazione binaria, avendo quindi ogni dataset suddiviso in due parti: una parte contenente i dati per l'addestramento del modello, mentre l'altra contenente i dati per il testing del modello. Nella classificazione binaria le etichette assegnabili sono solamente due, 1 oppure 0, che nel nostro caso corrispondono alla presenza o assenza di un picco nel battito cardiaco.
L'obbiettivo di questo lavoro è stata la ricerca del migliore modello e delle sue parametrizzazioni in modo tale da ottenere prestazioni ottimali da questi, in particolare la metrica che si è cercato di massimizzare è stata l'accuratezza, la quale ci dice le previsioni effettuate correttamente dal nostro modello. Le parametrizzazioni testate sono state differenti utilizzando diverse possibili combinazioni in modo da esplorare un maggior numero di casi. In particolare gli iperparametri testati sono stati *Batch_Size, Kernel_Size, Pool_Size, Ottimizzatore, numero di filtri, numero di epoche, learning_rate e momentum* oltre al tipo di pooling utilizzato dove sono stati testati entrambi i possibili livelli *MaxPooling* e *AveragePooling*. L'analisi dei segnali fisiologici mediante l'utilizzo dell'*Intelligenza Artificiale* può essere di significativo aiuto nel velocizzare tutti quei processi che altrimenti richiederebbero un grande dispendio di risorse e tempo restituendo inoltre una maggiore accuratezza dei risultati. Grazie a questo è possibile ottenere migliori diagnosi di malattie ai pazienti consentendo inoltre un monitoraggio più efficace e preciso che permette di migliorare la salute e la qualità della vita dei pazienti.

Capitolo 2

Related Works

In questo capitolo, esploreremo le sfide associate all'analisi dei segnali fisiologici in assenza di modelli di *Machine Learning* e come l'adozione di tali modelli possa aiutare a superare queste difficoltà, offrendo benefici significativi per gli esseri umani. Attraverso una revisione delle precedenti ricerche correlate, metteremo in luce le problematiche che emergono quando ci si affida esclusivamente a metodologie tradizionali di elaborazione dei segnali fisiologici. Inoltre, esamineremo le motivazioni che hanno guidato la scelta di adottare un approccio basato su modelli di *Machine Learning* in questo lavoro. Affronteremo anche le questioni legate alla mancata ottimizzazione di tali modelli e come ciò possa influenzare negativamente i risultati finali.

2.1 Utilizzo di modelli di Machine Learning per l'analisi di segnali fisiologici

L'utilizzo di modelli di *Machine Learning* per l'analisi di segnali fisiologici si rende necessario quando questi possono provenire da fonti molto diverse tra loro come, ad esempio, i segnali per l'analisi dell'interazione cervello-cuore [1], come già precedentemente introdotto, nel quale questi provengono da encefalogramma, elettrocardiogramma e attività respiratoria. Tra le numerose avversità dell'analisi di segnali fisiologici possiamo quindi individuare:

- **Complessità dei segnali:** i dati raccolti dalle varie fonti possibili possono essere estremamente complessi, il che renderebbe il processo di analisi e individuazione di pattern molto dispendioso e difficilmente praticabile. La complessità dei segnali è notevolmente aumentata dalla loro dipendenza dal soggetto preso in esame.

- **Presenza di inteferenze:** i segnali possono essere facilmente soggetti a interfenze o rumore dovuti all'ambiente circostante e alla sensibilità degli stumenti utilizzati. Basti pensare che nell'analisi *Brain-Heart Interaction* anche solo un

movimento degli occhi durante la misurazione può comportare la presenza di interferenza e rumore.

- **Accuratezza dei risultati:** la complessità dei segnali e la difficoltà nella loro processazione si ripercuote sulla qualità del risultato finale. Questo aspetto è fondamentale nel campo medico, nel quale le prestazioni devono essere sempre necessariamente elevate ammettendo una tolleranza agli errori molto bassa.

I problemi precedentemente elencati possono essere risolti grazie all'utilizzo di modelli di *Machine Learning* che permettono l'analisi di segnali molto complessi, l'individuazione di pattern non evidenti in una analisi manuale, gestendo correttamente rumori e interferenze introdotti in fase di raccolta dei dati e ottenendo un'accuratezza elevata come risultato finale. Possibili applicazioni, come il rilevamento dello stress [2] o la previsione di possibili malattie cardiache [5] possono essere usate come supporto per i medici per il monitoraggio e la prevenzione di malattie molto gravi, tutto ciò utilizzando modelli di *Machine Learning* con una accuratezza superiore al 95% e prossima al 99% che fa ben sperare per un loro futuro sviluppo e integrazione della vita di tutti i giorni.

2.1.1 Applicazioni per il riconoscimento dell' HRV da EEG

Questo studio fa riferemento al riconoscimento dell'*Heart Rate Variability* da *Elettroencefalogramma* tramite modelli di *Machine Learning*. Questo tipo di approccio è già stato studiato in alcuni casi precendenti ottenendo risultati interessanti tra i quali:

- **Sviluppo di un modello per la rilevazione di malattie:** sviluppare metodi non invasivi per la rilevazione delle malattie è una prospettiva fondamentale in campo medico. Per una migliore diagnosi è necessario rilevare dati proventienti da fonti differenti ,come l'ECG per il cuore e l'EEG per il cervello, e dover analizzare queste senza un supporto di un modello che individui pattern o anomalie all'interno dei dati è un grande problema. Per questo l'*Artificial Intelligence* ha trovato ampio spazio e applicazione nella predizione e rilevazione di eventuali malattie in un paziente [7].

- **Sviluppo di sistemi per la previsione di crisi epilettiche:** l'epilessia è una grave malattia che impatta la qualità del resto della vita di un paziente. I segnali biomedici coinvolti maggiormente nel monitoraggio di un paziente epilettico sono ECG ed EEG, oltre a il PPG o fotopletismografia. L'HRV può essere utilizzato per monitorare lo stato di salute di un paziente e come parametro predittore di possibili convulsioni. Questo parametro è fortemente influenzato dall' EEG dato che il compito della regolazione del battito cardiaco spetta al sistema nervoso. Risulta importante lo sviluppo di sistemi indossabili, sempre a disposizione del paziente, basati su algoritmi di *Intelligenza Artificiale* che permettano, tramite l'analisi dei segnali fisiologici misurati, la rilevazione

e la classificazione secondo una classe di rischio di possbibili crisi epilettiche future, per poter permettere al paziente di prendere le prevenzioni del caso ed avere il tempo di recarsi all'ospedale [12].

- **Sviluppo di modelli per la rilevazione personalizzata dello stress:** lo stress si riferisce all'insieme di riposte messe in pratica dal corpo umano per il superamento di situazioni pericolose. Quando però lo stress, soprattutto facendo riferimento a quello psicologico ed emotivo, si presenta in livelli troppo elevati ci possono essere conseguenze dannose per il nostro corpo, quando lo stress è prolungato nel tempo o addirittura cronico.
  Una pratica comune è il rilevamento dello stress nei pazienti tramite l'utilizzo di questionari, metodo facile da implementare ma soggetto ad imprecisioni. Un sistema basato sullo sviluppo di un modello di *Machine Learning* che analizzi i segnali biologici coinvolti nello stress, come EEG , ECG e HRV può portare ad effettuare analisi molto dettagliate e più precise rispetto all'approccio tradizionale, offrendo una diagnosi personalizzata al paziente mirata alla rilevazione di stress ed ansia per prevenire che queste condizioni portino allo sviluppo di problemi di salute di natura cronica [3].

- **Sviluppo di un modello per il riconoscimento delle emozioni:** oltre al settore medico, la stima e il riconoscimento delle emozioni, trova applicazioni in numerosi altri settori come il marketing o l'istruzione.
  Un primo approccio a questo campo può essere l'analisi di espressioni facciali, postura, comportamento e voce. Questo metodo risulta di facile applicazione essendo ossservabile dall'esterno, tuttavia ha un difetto che sta nella falsificabilità, un soggetto potrebbe voler esprimere altre emozioni intenzionalmente. Il secondo metodo mira a risolvere questo problema utlizzando indici fisiologici come ECG, EEG, HRV e EMG. Questi segnali biologici hanno il vantaggio di essere difficilmente modificabili arbitrariamente da un umano. Lo sviluppo di un modello di *Machine Learning* che utilizza questo secondo approccio restituisce una stima e classificazione delle emozioni in maniera oggettiva e con una maggiore accuratezza [9].

2.2 Ottimizzazione degli iperparametri per un modello

Una volta scelto il modello da utilizzare per l'analisi dei segnali fisiologici è importante testare le varie parametrizzazioni offerte. La mancata ottimizzazione degli iperparametri potrebbe portare a problematiche come:

- **Prestazioni non ottime:** l'utilizzo di iperparametri che non si adattano allo specifico caso di studio non permette al modello di performare come dovrebbe, ottenendo come risultato delle prestazioni che sono soltanto subottimali ma

che potrebbero essere migliorate, anche notevolmente, grazie al tuning degli iperparametri.

- **Overfitting e underfitting:** questi problemi sono causati quando un modello si adatta molto bene ad un set di dati memorizzando così il relativo rumore e i pattern del dataset, causando un calo delle prestazioni quando al modello verranno presentati dati a lui sconosciuti. Nel caso opposto, utilizzando un modello troppo semplice per quel dataset, si verificherà il problema dell'underfitting.[4]

- **Dispendio di risorse:** una mancata o cattiva ottimizzazione dei parametri, potrebbe portare un modello, che ha già una sua pesantezza, a dover necessitare di un maggior numero di risorse a causa della sua complessità computazionale, portando così ad un utilizzo inefficace dei mezzi a disposizione, con particolare attenzione al tempo necessario per l'addestramento del modello.

La ricerca delle migliori parametrizzazioni da utilizzare per il modello è quindi necessaria se si vogliono ottenere le migliori prestazioni.[10][8][11]
Questo perchè, nonostate ogni modello abbia la propria parametrizzazione di default, questa potrebbe non essere ottimale per il caso di studio su cui si sta lavorando. Ogni dataset è diverso e presenta dati tra loro differenti e necessiterà di un suo specifico insieme di iperparametri per ottenere le performance ottimali.
Questo processo può essere svolto tramite una ricerca manuale, utile quando abbiamo pochi iperparametri con un numero limitato di valori da dover testare, oppure svolta da apposite funzioni, che operano automaticamente in maniera differente nella ricerca del migliore insieme di parametri, come possono essere *GridSearchCV, RandomSearchCV, Asynchronous Successive Halving Algorithm (ASHA)* o *Bayesian Optimization*.

Capitolo 3

Design e Implementazione

Nel capitolo viene affrontato l'approccio utilizzato in questo studio, descrivendone la sua implementazione, per riuscire ad affrontare i problemi messi in luce precedentemente e i possibili casi d'uso del sistema.

3.1 Design

In questa sezione sono messe in risalto le scelte adottate per il miglioramento prestazionale di un modello per l'analisi di segnali fisiologici.

3.1.1 Classificazione

Nel *Machine Learning* la classificazione consiste nel predire la classe, anche chiamata etichetta, di appartenenza per ciascuna istanza di dati. Questo permette al modello, dopo la fase di addestramendo sul training set, di classificare con una certa accuratezza nuove istanze di dati sconosciute ad esso.
Nel caso studiato il problema che affrontiamo appartiene alla categoria della classificazione binaria nella quale abbiamo solo due classi da assegnare, *1* per le istanze che corrispondo ai cosiddetti *peak* ovvero i picchi del battito cardiaco rilevati nell'ECG, *0* per le istanze che identificano i *non-peak*.
Nel dataset preso in considerazione i segnali fisiologici misurati sono divisi in 150 features rappresentate da un numero crescente da 1 fino a 150.
I modelli di classificazione permettono di affrontare il problema della classificazione dei dati.

3.1.2 Modelli utilizzati

Nel corso dello studio sono stati utilizzati principalmente tre tipi di modelli differenti tra loro:

- **Convolutional Neural Network:** questo tipo di modello è molto simile all' *Artificial Neural Network*, modello composto da molti nodi, detti neuroni, ca-

paci di apprendere attraverso l'addestramento. Sono solitamente utilizzate con immagini come dati di input e sono composti da diversi livelli che possono essere, *input layer*, *convolutional layer* che permette di realizzare la peculiarità di questo modello, ovvero l'operazione di convoluzione tra i dati in input e specifici filtri, *pooling layer* livello che è sempre sottostante al livello di convoluzione, il cui scopo è ridurre la dimensione della feature map, ovvero l'output del convolutional layer, *fully-connected layers* composti dai neuroni che permetteranno l'apprendimento del modello, e infine l'*output layer*[6].
Nel lavoro sono stati utilizzati due tipi di *CNNs*:

  - **CNN1D:** i dati in input sono presi come array, quindi vettori di elementi composti soltanto da una singola dimensione.
  - **CNN2D:** i dati in input sono matrici. La convoluzione non sarà più quindi realizzata tra vettori monodimensionali ma tra matrici che sono composte da due dimensioni, con un aumento della complessità dell'operazione.

- **Learning Shapelets:** modello specifico per le *time-series*, ovvero una sequenza di dati ordinati secondo il tempo. Il modello utilizzato si concentra sull'individuazione di *shapelets*, ovvero forme e pattern particolari all'interno delle serie temporali che possono portare all'assegnazione di una classe o di un'altra in un problema di classificazione.

3.1.3 Pre-processing dei dati

Le tecniche di *pre-processing* permettono la preparazione dei dati in modo tale che siano nella forma ottimale per poter essere utilizzati dal modello. Un mancato o scorretto pre-processing può portare a un degrado delle prestazioni, situazione che vogliamo sempre evitare.
La tecnica di pre-processing utilizzata è stata una, **StandardScaler** che permette la standardizzazione dei dati, cioè trasformarli in modo che si ottenga una media pari a zero($\mu = 0$) e una deviazione standard pari a 1($\sigma = 1$).
Dato x dato non standardizzato e y dato ottenuto dopo la standardizzazione, questo processo avviene secondo il seguente calcolo:

$$y = \frac{x - \mu}{\sigma}$$

3.1.4 Ottimizzazione degli iperparametri

Come già precedentemente discusso, l'ottimizzazione degli iperparametri si rende necessaria quando vogliamo ottenere le massime prestazioni dal nostro modello.
È importante sottolineare la differenza che c'è tra parametri e iperparametri:

- I **parametri** sono dei valori che vengono appresi direttamente dal modello durante il processo di addestramento.

- Gli **iperparametri** sono configurazioni esterne al modello e non possono essere ricavati dai dati, ma devono essere specificati prima dell'addestramento.

I vari iperparametri testati sono stati diversi, per esplorare un maggiore numero di combinazioni che potessero migliorare le performance, utilizzando una serie di *for* innestati, in modo da simulare il *GridSearchCV*, dato che alcuni iperparametri erano incompatibili con l'uso di questa funzione.
Tra i vari iperparametri testati si hanno:

- **Batch_Size:** stabilisce il numero di samples utilizzati ad ogni iterazione nel processo di addestramento. Il set di dati verrà dunque diviso in batch ciascuno di dimensione batch_size.

- **Kernel_Size:** nell'operazione di convoluzione della rete neurale si ha il filtro che, scorrendo sull'input, moltiplicherà i valori estratti con i valori dei pesi del filtro stesso. Il kernel_size definisce la grandezza di questo filtro. Filtri più grandi consentono di avere una finestra maggiore sull'input, mentre filtri più piccoli possono essere usati per cogliere maggiormente i dettagli nell'input.

- **Pool_Size:** è un iperparametro facente parte del layer di pooling, che definisce di quanto viene ridotta questa feature map. Ad esempio con una feature map di dimensione 6x6 che viene data in input al layer di pooling con pool_size di 2, verrà applicata una matrice 2x2 alla mappa che dimezzerà la dimensione di quest'ultima.

- **Filters** che rappresenta il numero di filtri utilizzati per la convoluzione.

- **Epochs** che ci dice il numero di iterazioni effettuate durante il processo di addestramento.

- **Learning_rate** utile all'ottimizzatore SGD, indica la dimensione dei passi effettuati durante la discesa del gradiente.

- **Momentum** parametro di SGD utile ad accelerare la convergenza al minimo globale superando eventuali minimi locali.

Oltre agli iperparametri sono stati effettuati ulteriori cambiamenti andando a testare modifiche che sono parte integrante del modello come:

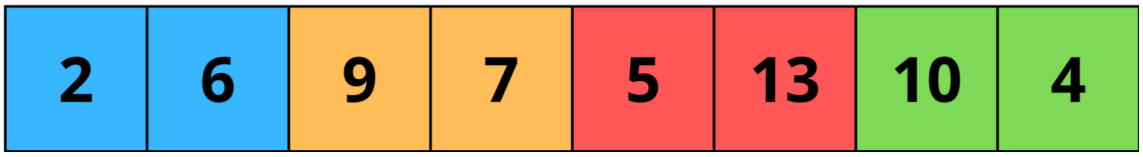
- **Algoritmo di ottimizzazione:** permette di migliorare le prestazioni del modello minimizzando la funzione di perdita grazie all'aggiornamento dei pesi dei filtri, permettendo al modello di migliorare l'apprendimento.
  In particolare gli ottimizzatori utilizzati sono stati:

  - **Stochastic Gradient Descent (SGD):** il quale obbiettivo è la ricerca del punto di minimo globale di una funzione partendo da un punto randomico di essa. Una volta trovato il minimo questo verrà utilizzato per l'aggiornamento dei pesi.
  - **Adaptive Moment Estimation (ADAM):** in questo algoritmo i pesi vengono aggiornati grazie alla stima del momento primo e secondo dei gradienti e all'aggiornamento del tasso di apprendimento che avviene in modo adattivo in base ai valori precendenti dei gradienti.

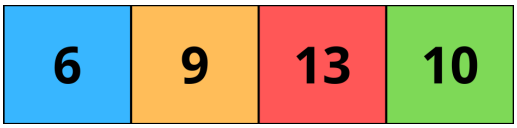
- **Metodo di Pooling:** come già precedentemente affermato il *pooling layer* permette di ridurre la dimensione della *feature map*, output del *convolutional layer*.
  Nel livello si distinguono principalmente due metodi di pooling:

  - **MaxPooling:** viene preso il valore massimo all'interno del blocco della feature map considerato.
  - **AveragePooling:** viene effettuata la media dei valori presenti all'interno del blocco della feature map considerato.

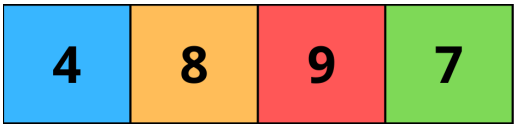
  Questi due metodi di pooling considereranno diversamente le informazioni della feature map dato che nel MaxPooling quelle ritenute meno importanti vengono ignorate prendendo in cosiderazione solo quella di maggiore rilevanza, mentre nell'AveragePooling si tiene conto di tutte le informazioni presenti nel blocco.



(a) Possibile feature map di una CNN 1D



(b) Metodo MaxPooling



(c) Metodo AveragePooling

Figura 3.1: Confronto dell'applicazione dei due metodi su una feature map con pool_size=2

3.1.5 Metriche per l'analisi prestazionale del modello

Le metriche sono fondamentali per valutare le performance ottenute dal modello e per permettere un confronto tra i vari modelli e i diversi iperparametri utilizzati.
Per la comprensione delle metriche dobbiamo definire:

- **True Positives (TP):** numero di casi in cui il modello ha previsto *correttamente* che l'istanza appartiene alla classe *positiva*.
- **True Negatives (TN):** numero di casi in cui il modello ha previsto *correttamente* che l'istanza appartiene alla classe *negativa*.
- **False Positives (FP):** numero di casi in cui il modello ha previsto *erroneamente* che l'istanza appartiene alla classe *positiva*.
- **False Negatives (FN):** numero di casi in cui il modello ha previsto *erroneamente* che l'istanza appartiene alla classe *negativa*.

Le metriche utilizzate possono essere espresse grazie a questi quattro valori, e in particolare si ha:

- **Accuracy:** rappresenta il numero di previsioni corrette effettuate dal modello rispetto al totale delle previsioni.

  $$\textbf{Accuracy} = \frac{\textbf{Numero di previsioni corrette}}{\textbf{Numero totale di previsioni}}.$$

  In alternativa può essere espressa in funzione dei quattro valori definiti prima come:

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

- **Recall:** è il rapporto tra il numero di predizioni positive fatte correttamente e tutte le predizioni positive che potevano essere fatte.

  $$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP}+\textbf{FN}}.$$

- **Precision:** è il rapporto tra il numero di predizioni positive fatte correttamente e tutte le istanze predette come positive.

  $$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP}+\textbf{FP}}.$$

- **F1_Score:** combina Recall e Precision facendone la media armonica.

  $$\textbf{F1\_score} = \textbf{2} \times \frac{\textbf{Recall}\times\textbf{Precision}}{\textbf{Recall}+\textbf{Precision}}.$$

3.2 Implementazione

Lo studio svolto si basa sull'analisi e la classificazione di segnali fisiologici rilevati grazie ad appositi strumenti utilizzati sul paziente.
Il settore principale in cui l'approccio può avere una maggiore applicazione è quello medico, come già detto già in precedenza.

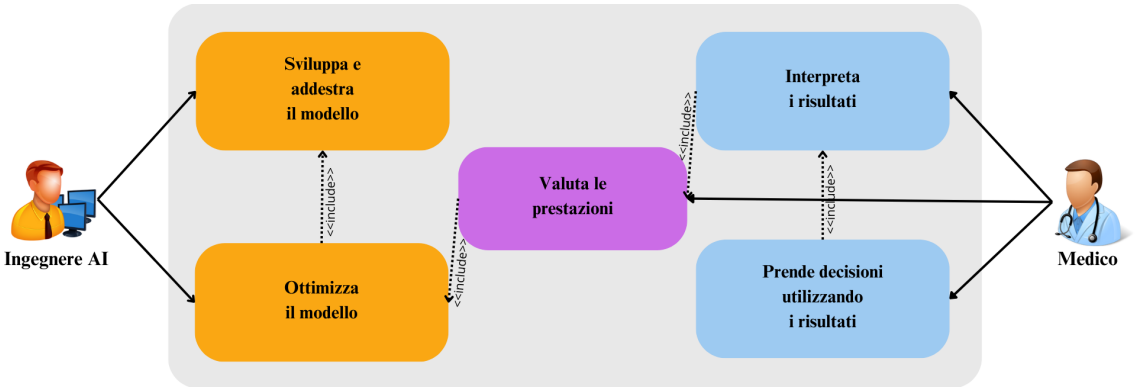


Figura 3.2: Diagramma dello Use case

3.2.1 Use case

Nella Figura 3.2 è rappresentato un passibile caso d'uso in campo medico. Possiamo notare la presenza di due attori:

- **Ingegnere AI:** attore secondario che si occupa dello sviluppo, addestramento e ottimizzazione del modello, mantenedosi sempre in contatto con il medico per assicurarsi che il sistema sviluppato soddisfi le richieste.
  Più nel dettaglio l'ingegnere AI si occupa di:
  - **Sviluppare e addestrare il modello:** questo è uno dei compiti principali dell'ingegnere AI che deve scegliere il modello che più si adatta al caso preso in analisi scegliendo tra vari tipi di modelli come possono essere le *Convolutional Neural Networks* o *Learning Shapelets*. Una volta sviluppato il modello è suo compito realizzare anche la fase di addestramento in modo tale che le performance del modello siano sempre le massime ottenibili.
  - **Ottimizzare il modello:** non sempre i risultati ottenuti sono quelli sperati. Per rimediare a questo l'ingegnere AI deve procedere all'ottimizzazione del modello osservando i risultati ottenuti, anche grazie alla collaborazione del medico che può restituire eventuali feedback.
- **Medico:** attore principale dato che è colui a cui è destinato il modello sviluppato e svolge le seguenti azioni:
  - **Interpretare i risultati:** per prendere le corrette decisioni è necessario interpretare i risultati che il modello ha prodotto. Dopo la classificazione dei segnali fisiologici, con una attenta esaminazione dei risultati è possibile riuscire a individuare eventuali pattern o incongruenze che possono portare ad una diagnosi più completa e corretta.
  - **Prendere decisioni utilizzando i risultati:** dopo l'interpretazione dei risultati il medico si deve occupare del prendere decisioni in base a ciò che ha ottenuto. Queste decisioni possono coinvolgere eventuali diagnosi di

malattie ad un paziente o eventuali terapie che il paziente dovrà effettuare. In ogni caso è necessario che in ogni momento siano prese le migliori decisioni possibili.

– **Valutazione delle prestazioni:** come utilizzatore finale del prodotto è necessario che il medico si occupi anche della valutazione delle performance del modello, sulla base di risultati ottenuti in passato e di come questi lo hanno aiutato. Questo procedimento è necessario per restituire all'ingegnere AI un feedback in modo tale che esso possa, quando possibile, ottimizzare il modello ed eventualmente incrementarne le prestazioni.

Capitolo 4

Case Study

4.1 Dataset

Il dataset preso in considerazione per questo lavoro deriva da uno studio realizzato in collaborazione tra l'*Università di Pisa* e il *Politecnico di Milano* sull'interazione cervello-cuore sotto stress termico.

4.1.1 Descrizione del dataset

Lo studio è stato condotto inzialmente su 32 giovani adulti destrimani senza malattie neurologiche, cardiovascolari o respiratorie.
Non tutti i dati raccolti dai soggetti sono stati presi in considerazione, in particolare:

- I dati di tre soggetti sono stati scartati a causa della presenza di anomalie o interferenze nei loro dati fisiologici.
- I dati di tre soggetti sono stati scartati a causa del ritiro precoce della mano dall'acqua fredda.

I dati raccolti per ogni soggetto comprendevano *elettroencefalogramma (EEG)* ad alta densità a 128 canali, grazie al caschetto in Figura 4.1 , *attività respiratoria* ed *elettrocardiogramma (ECG)* a una derivazione campionati a 500 Hz.
Prima dell'acquisizione dei dati i soggetti sono stati fatti sedere su una sedia per garantire la stabilizzazione emodinamica. Dopodichè la raccolta dei dati vera e propria compredeva:

1. Stato di riposo per tre minuti.
2. Test di pressione a freddo fino a 3 minuti.
3. Ritiro della mano dall'acqua ghiacciata e ulteriore riposo.

Durante l'acquisizione dei dati, al soggetto, è stato chiesto di tenere gli occhi chiusi per evitare anche la minima interferenza nella misurazione, guidando poi la sua mano sinistra nel secchio di acqua ghiaccato con una temperatura compresa tra 0°C e 4°C[1].

Tabella 4.1: Sintesi del dataset

Numero di soggetti	26 (13 uomini e 13 donne)
Età dei soggetti	21-41 anni (mediana 27 anni)
Segnali fisiologici acquisiti	EEG, ECG, Attività respiratoria
Frequenza di campionamento	500Hz

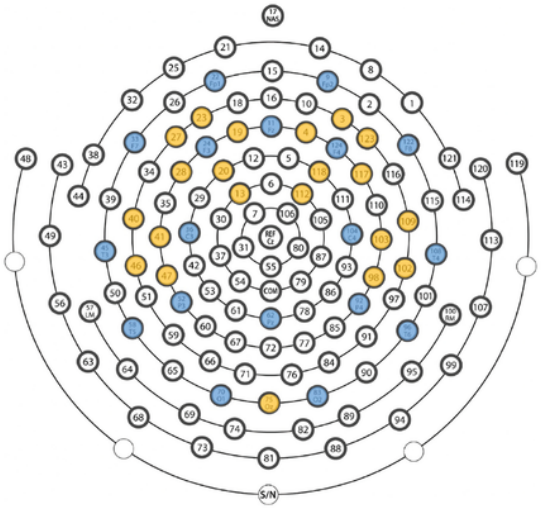


Figura 4.1: Caschetto EEG utilizzato per la raccolta dei dati

4.1.2 Processing dei dati EEG

Il processing dei dati EEG è stato eseguito secondo i seguenti passi:

- **Filtraggio passa-banda**: I dati EEG sono stati filtrati utilizzando un filtro passa-banda(filtro di Butterworth di ordine 4) con frequenze di taglio tra 0,5 e 45 Hz per rimuovere frequenze indesiderate [1].

- **Selezione dei canali**: Sono stati considerati solo i canali EEG situati sul cuoio capelluto (97 dei 128 canali) per evitare la contaminazione dei dati non neurali [1].

- **Rimozione delle interferenze di movimento**: Le interferenze di movimento di grandi dimensioni sono state eliminate utilizzando l'analisi delle componenti indipendenti potenziate dalle wavelet [1].

- **Riconoscimento degli artefatti oculari e cardiaci tramite ICA**: È stata eseguita una *Indipendent Component Analysis (ICA)* per identificare e rimuovere i movimenti oculari e le interferenze cardiache dai dati EEG. L'ECG è stato incluso come input aggiuntivo per migliorare la ricerca di queste interferenze [1].

- **Rimozione dei componenti ICA contaminati**: I componenti ICA con movimenti oculari e artefatti cardiaci sono stati identificati visivamente e impostati a zero per ricostruire la serie EEG [1].
- **Rilevamento dei canali contaminati**: I canali EEG sono stati contrassegnati come contaminati se la loro attività superava 3 deviazioni standard dalla media di tutti i canali. I canali contaminati sono stati sostituiti con l'interpolazione del vicino [1].
- **Rireferenziazione dei canali**: I dati sono stati rireferenziati utilizzando una media comune, più appropriata per lo studio eseguito [1].
- **Selezione dei canali e calcolo dello spettrogramma**: Un sottoinsieme di 64 canali è stato selezionato per ridurre la ridondanza. Lo spettrogramma EEG è stato calcolato utilizzando la trasformata di Fourier. I calcoli sono stati eseguiti attraverso una finestra temporale scorrevole di 2s con una sovrapposizione del 50%, ottenendo una risoluzione dello spettrogramma di 1s e 0,5 Hz [1].
- **Integrazione in bande di frequenza**: Le serie temporali sono state integrate in cinque bande di frequenza [1]:
  - **Delta:**1-4 Hz.
  - **Theta:**4-8 Hz.
  - **Alfa:**8-12 Hz.
  - **Beta:**12-30 Hz.
  - **Gamma:**30-45 Hz.

4.1.3 Processing dei dati ECG

Per il processing dei dati ECG sono state eseguite invece le seguenti azioni:

- **Filtraggio passa-banda**: Le serie temporali ECG sono state filtrate in banda passa-banda utilizzando un filtro di Butterworth di ordine 4, tra 0,5 e 45 Hz, per rimuovere frequenze indesiderate e ridurre il rumore [1].
- **Identificazione dei picchi R**: I picchi R delle onde QRS sono stati identificati prima tramite un processo automatizzato, seguito da un'ispezione visiva dei rilevamenti errati e dalla correzione automatica finale dei rimanenti errori di rilevamento o battiti cardiaci ectopici [1].
- **Correzione manuale e automatica dei picchi R**: Tutti i picchi rilevati sono stati ispezionati visivamente sull'ECG originale, insieme all'istogramma degli intervalli tra i battiti. Dove necessario sono state applicate prima correzioni manuali e successivamente correzioni automatiche [1].

4.2 Librerie utilizzate

Tutto il codice utilizzato è stato realizzato con Python, grazie a librerie specifiche per lo sviluppo, addestramento e l'ottimizzazione di modelli di *Machine Learning* e per la realizzazione di grafici per l'analisi prestazionale dei modelli. Di seguito sono riportate le librerie e le eventuali funzioni utilizzate di esse:

- **tensorflow:** è una libreria open-soruce sviluppata da *Google* per lo sviluppo, addestramento e ottimizzazione di modelli di *Machine Learning*.
  Questa libreria è fondamentale dato che permette l'utilizzo della libreria **Keras**, con quest'ultima che fornisce una interfaccia ad alto livello per Tensorflow riguardante l'utilizzo di modelli di reti neurali. Di queste librerie sono state utilizzate le seguenti funzionalità:
  - **SGD:** funzione che riproduce *Stochastic Gradient Descent* come alogoritmo di ottimizzazione.
  - **Adam:** permette l'utilizzo dell'algortimo di ottimizzazione *Adaptive Moment Estimation*.
  - **Sequential:** funzione che permette di definire la struttura di un modello di rete neurale come *sequenza lineare di strati*, ovvero dove i layer sono posizionati come in una pila e ciascuno di essi ha un singolo input e un singolo output.
  - **Conv1D:** utilizzato per lo strato convoluzionale unidimensionale.
  - **Conv2D:** utilizzato per lo strato convoluzionale bidimensionale.
  - **MaxPooling1D:** rappresenta il layer di pooling con metodo Max per una 1DCNN.
  - **AveragePooling1D:** rappresenta il layer di pooling con metodo Average per una 1DCNN.
  - **MaxPooling2D:** rappresenta il layer di pooling con metodo Max per una 2DCNN.
  - **Flatten:** funzione utilizzata per aggiungere uno strato di appiattimento che permette di trasformare l'input da un array multidimensionale ad uno unidimensionale.
  - **Dense:** utilizzata per aggiungere gli strati in cui sono inseriti i neuroni che permetteranno l'apprendimento del modello.
  - **ReLu:** rappresenta la funzione di attivazione. La funzione ReLu è una funzione molto semplice definita come $f(x) = max(0, x)$, ovvero si ottiene 0 se abbiamo una x negativa, altrimenti si ottiene x. Un'altra funzione di attivazione molto utilizzata è Sigmoid che è definita come $\sigma(x) = \frac{1}{1+e^{-x}}$ producendo quindi valori compresi tra 0 ed 1.

- **scikit-learn:** permette l'utilizzo di funzionalità necessarie per il processamento dei dati, implementazione e addestramento dei modelli e la valutazione delle prestazioni di esse.
  Per questa libreria sono state utilizzate le funzionalità:
  - **StandardScaler:** funzione di pre-processing dei dati che permette la standardizzazione di quest'ultimi.
- **tslearn:** è un libreria specializzata nell'analisi di serie temporali, e fornisce funzionalità che permettono l'elaborazione e l'analisi di esse oltre alla creazione e la valutazione di modelli basati su time series. Le funzionalità messe a disposizione che sono state usate sono le seguenti:
  - **to_time_series_dataset:** permette di convertire un array di serie temporali in un formato compatibile con *tslearn*.
  - **TimeSeriesScalerMeanVariance:** funzione utilizzata per la standardizzazione di time series.
  - **LearningShapelets:** permette di costruire un modello di *Learning Shapelets* già descritto in precedenza.
- **numpy:** libreria che consente la manipolazione di array multidimensionali.
- **pandas:** libreria per la gestione dei dati, anche in grande quantità come nel caso di DataFrame.
- **matplotlib:** libreria utilizzata per la creazione di grafici per la visualizzazione dei risultati in modo compatto ed efficente.

Capitolo 5

Risultati Sperimentali

Nel seguente capitolo, sono esposti i risultati sperimentali derivanti dai metodi precedentemente introdotti. Questa fase è cruciale per valutare le prestazioni e l'efficacia delle tecniche proposte. I risultati ottenuti sono proposti sia in forma tabellare che per via grafica. In particolare sono stati utilizzati due tipi di grafici:

- **Istogrammi.**
- **Box Plot:** in questi grafici abbiamo una scaltola i cui bordi inferiori e superiori rappresentano rispettivamente il 25° e75° quantile. La lunghezza della scatola è chiamata range interquartile (IQR) ed è data dalla differenza tra i due quantili. All'interno della scatola avremo una linea che rappresenta la mediana dei dati e ci indica che il 50% dei dati si trova sopra e l'altro 50% sotto. Dalla scatola partono delle linee che sono chiamati baffi e hanno una lunghezza pari a 1,5 volte l'IQR. Se i dati non arrivano alla fine dei baffi allora questi si estenderanno fino ai valori massimi o minimi. Se ci sono invece dati che superano la dimensione dei baffi allora questi sono definiti come outliers.

5.1 Ottimizzazione di una Convolutional Neural Network Unidimensionale

Il primo modello preso in considerazione è stata una *CNN1D*, già discussa precedentemente, su cui è stato realizzato un processo di ottimizzazione, tramite la ricerca dei migliori iperparametri, algoritmo di ottimizzazione e metodo di pooling, per ottenere un miglioramento delle prestazioni. Inizialmente la metrica presa in cosiderazione per il confronto delle performance è stata l'*accuracy*, già definita in precedenza.

5.1.1 Ottimizzazione dell'iperparametro batch_size

Come prima fase di ottimizzazione del modello è stato scelto l'iperparametro *batch_ size*, andando a testare valori piccoli come **8** e **16**.

Tabella 5.1: Statistiche sull'accuratezza al variare di batch_size

Batch_size	**Max Accuracy**	**Min Accuracy**	**Media Accuracy**	**Dev. Std. Accuracy**
8	96.88%	30.00%	74.58%	14.93%
16	96.88%	30.00%	74.53%	15.96%

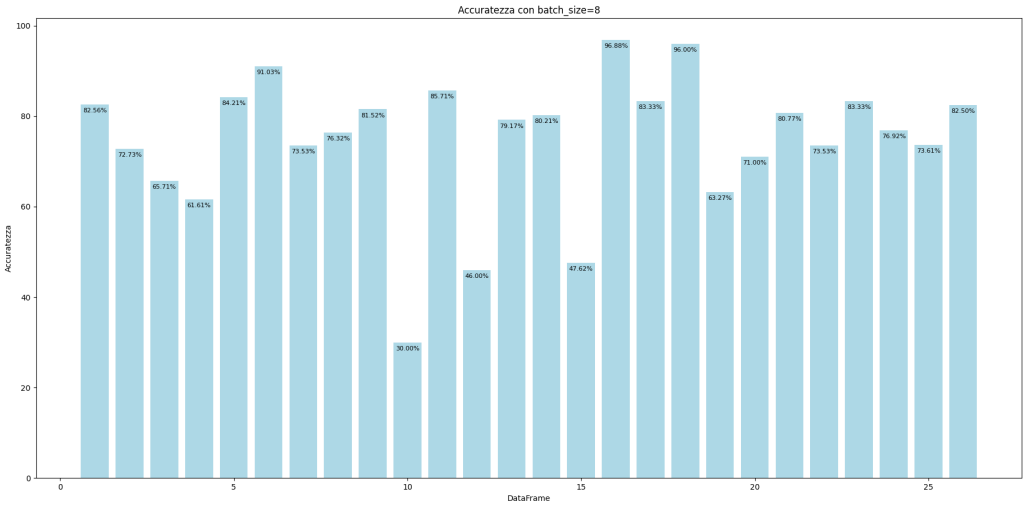


Figura 5.1: Istogramma dell'accuratezza con batch_size=8

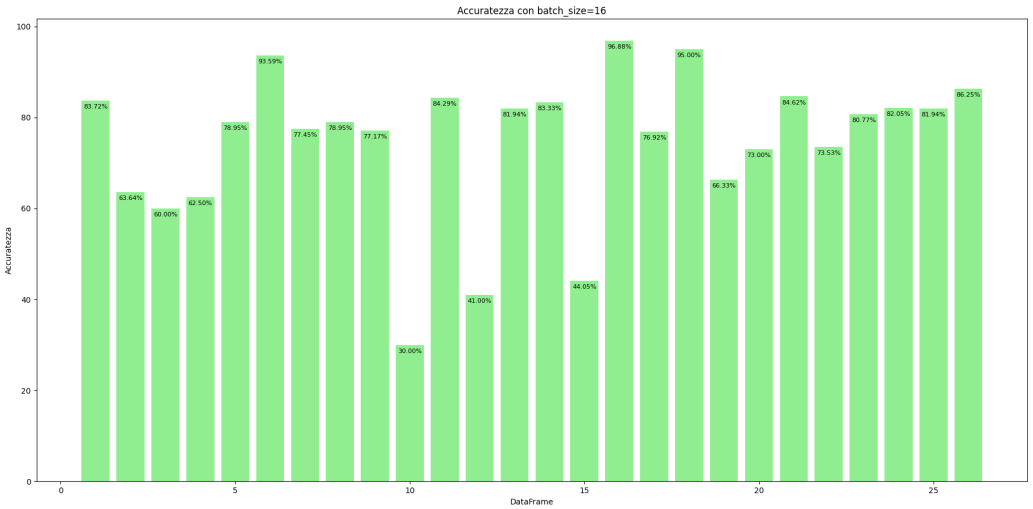


Figura 5.2: Istogramma dell'accuratezza con batch_size=16

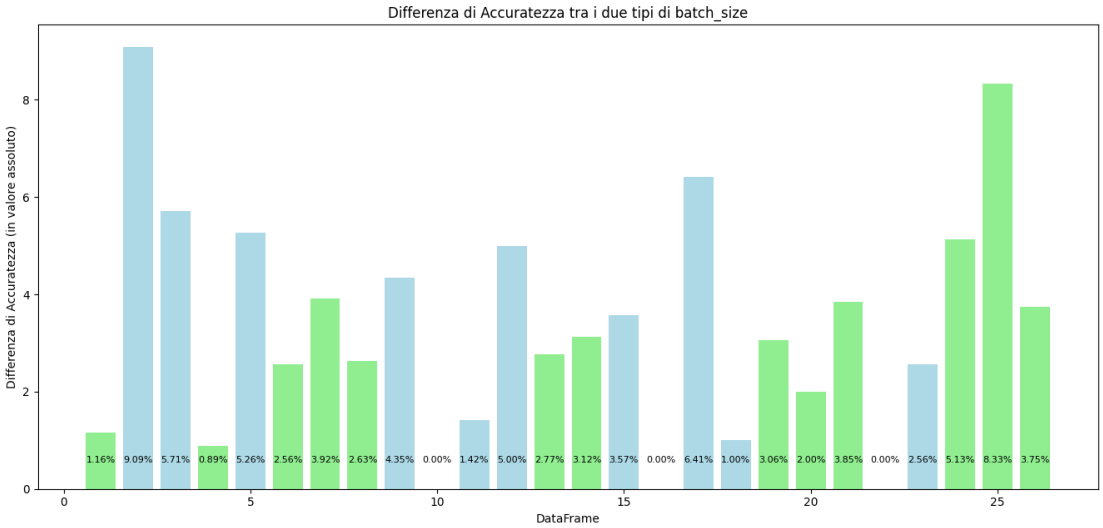


Figura 5.3: Istogramma della differenza di accuratezza

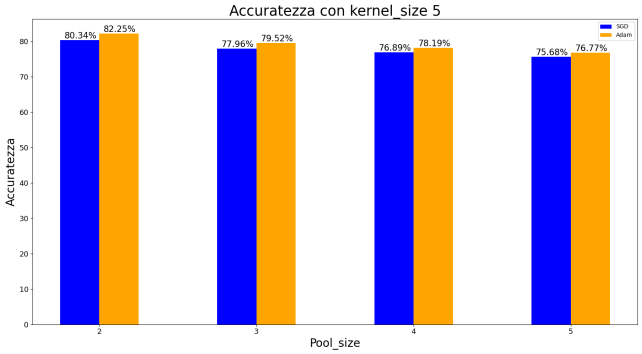
Possiamo notare come le prestazioni ottenute non sono sicuramente ottimali, ottenendo circa un 75% di accuratezza, con una sensibile differenza che tende a favore di un batch_size pari ad 8. Nelle successive fasi quindi vengono testate ulteriori parametrizzazioni per incrementare le performance fissando l'iperparametro testato in questa prima fase ad 8.

5.1.2 Ottimizzazione di Kernel_Size, Pool_Size e ottimizzatore

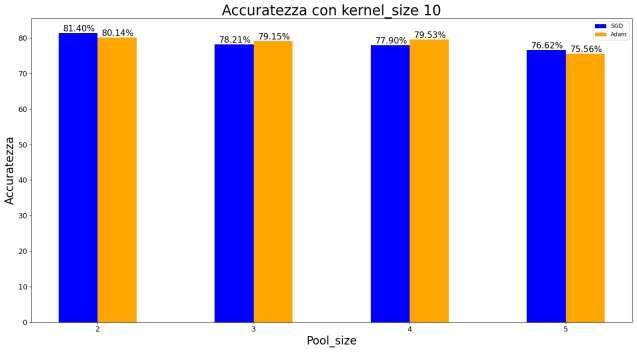
In questa seconda fase vengono valutate le prestazioni del modello andando a ricercare i migliori valori per:

- **Kernel_Size**. Per questo iperparametro sono stati valutati i valori 5, 10, 15, 20.
- **Pool_Size**. Per questo iperparametro sono stati valutati i valori 2, 3, 4, 5.
- **Ottmizzatore**. Sono usati 2 tipi di ottimizzatore: SGD e Adam.

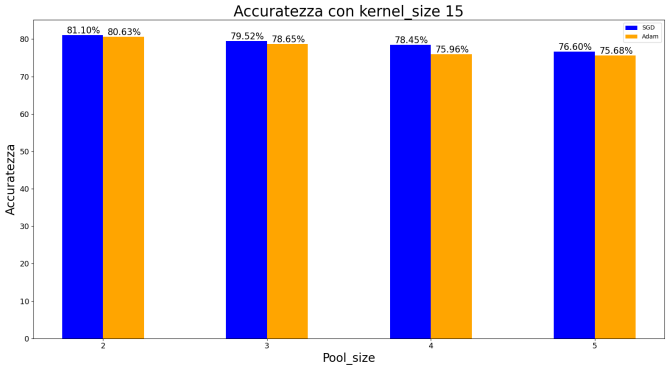
Otteniamo quindi un totale di 32 possibili combinazioni da esplorare per il miglioramento delle performance del nostro modello.



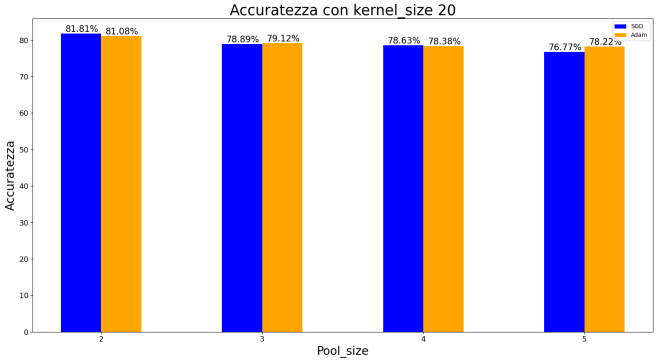
(a) Istogramma per il confronto dell'accuratezza con kernel size 5



(b) Istogramma per il confronto dell'accuratezza con kernel size 10



(c) Istogramma per il confronto dell'accuratezza con kernel size 15



(d) Istogramma per il confronto dell'accuratezza con kernel size 20

Figura 5.4: Istogrammi per il confronto dell'accuratezza

Tabella 5.2: Prestazioni del modello con algoritmo di ottimizzazione SGD

(a) **Media dell'accuracy**

	2	**3**	**4**	**5**
5	80.34%	77.96%	76.89%	75.68%
10	81.40%	78.21%	77.90%	76.62%
15	81.10%	79.52%	78.45%	76.60%
20	81.81%	78.89%	78.63%	76.77%

(b) **Dev. Std. dell'accuracy**

	2	**3**	**4**	**5**
5	16.18%	15.52%	16.71%	16.16%
10	15.26%	16.75%	15.68%	16.46%
15	15.35%	15.59%	17.07%	15.86%
20	15.12%	15.85%	16.77%	16.03%

Tabella 5.3: Prestazioni del modello con algoritmo di ottimizzazione Adam

(a) **Media dell'accuracy**

	2	**3**	**4**	**5**
5	82.25%	79.52%	78.19%	76.77%
10	80.14%	79.15%	79.53%	75.56%
15	80.63%	78.65%	75.96%	75.68%
20	81.08%	79.12%	78.38%	78.22%

(b) **Dev. Std. dell'accuracy**

	2	**3**	**4**	**5**
5	14.61%	16.35%	14.84%	16.18%
10	16.53%	16.41%	14.9%	17.45%
15	15.98%	15.49%	17.36%	17.77%
20	15.38%	15.05%	16.50%	14.24%

Osservando la Tabella 5.2 e la Tabella 5.3 e le Immagini 5.4 , possiamo notare che con entrambi gli algoritmi all'aumentare del pool_size l'accuratezza diminuisce indipendentemente dal valore di kernel_size, con una variazione che può arrivare anche a circa 6 punti percentuali a parità di kernel_size. Questo ci può suggerire che aumentando il livello di pooling si ha un'approssimazione maggiore dovuta alla riduzione superiore della feature map dato anche l'utilizzo di MaxPooling, che prende una decisione che verte sul massimo valore contenuto nella matrice di pooling invece che una media di questi valori come nell' AveragePooling.
Per SGD possiamo notare anche che avendo lo stesso pool_size all'aumentare del kernel_size aumenta, anche se sensibilmente, l'accuratezza. Non può essere affermata la stessa cosa per Adam dato che in questo caso non sembra seguire l'andamento

di SGD.
Esplorando queste possibili combinazioni possiamo osservare come le prestazioni siano incrementate notevolmente rispetto ai risultati che avevamo ottenuto inizialmente.

5.1.3 Valutazione delle prestazioni al variare del metodo di pooling

In precedenza è stato osservato come l'aumentare del pool_size potesse influenzare negativamente le performance del nostro modello. Data l'influenza di questo parametro è stato scelto di verificare se anche il metodo di pooling potesse influenzare le prestazioni della CNN. Per fare questo sono stati messi a confronto i due metodi *MaxPooling* e *AveragePooling*, restringendo i possibili valori degli iperparametri presi in considerazione precedentemente. In particolare:

- **Kernel_Size:** sono stati utilizzati i valori 5 e 20.
- **Pool_Size:** sono stati utilizzati i valori 2 e 3.
- **Ottimizzatore:** sono stati utilizzati sia SGD che Adam.

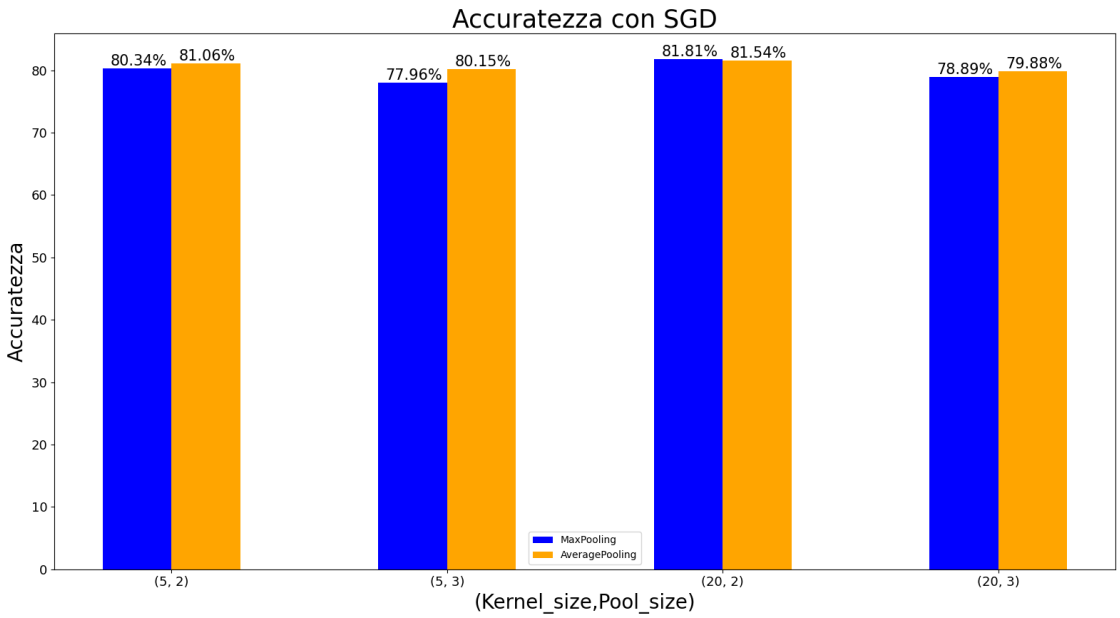


Figura 5.5: Accuratezza al variare dell'algoritmo di pooling con SGD

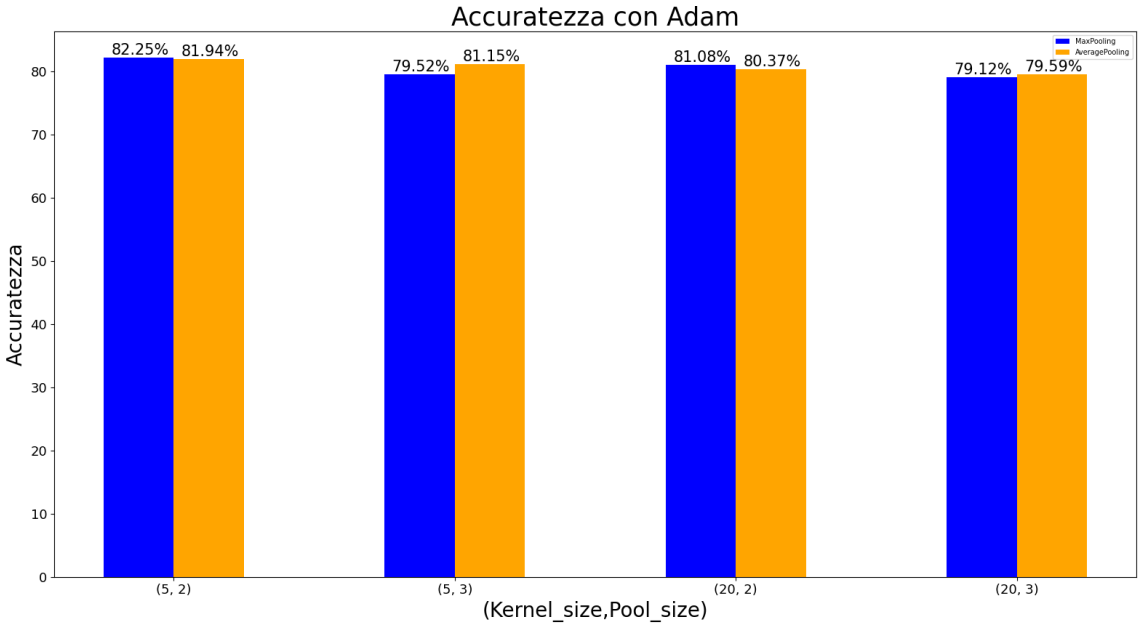


Figura 5.6: Accuratezza al variare dell'algoritmo di pooling con Adam

A differenza del parametro pool_size, non abbiamo una grande differenza se andiamo a confrontare le prestazioni ottenute con i due metodi di pooling. È possibile affermare che utilizzando AveragePooling si ha mediamente un' accuratezza superiore con le varie parametrizzazioni utilizzate, ma i valori massimi di accuratezza sono stati ottenuti utilizzando il metodo di MaxPooling.

5.1.4 Ottimizzazione di filters, epochs, learning_rate e momentum

Per proseguire con l'ottimizzazione del modello è necessario testare nuovi iperparametri e possibili combinazioni, mentre è stato utilizzato il metodo di MaxPooling con un pool_size pari a 2 con l'ottimizzatore SGD. Gli iperparametri testati sono stati:

- **Kernel_Size:** è stato scelto di testare nuovamente entrambi i valori per analizzarne nuovamente il comportamento.
- **filters:** sono stati utilizzati i valori 32, 64 e 128.
- **epochs:** sono stati testati i valori 50, 150 e 250.
- **learning_rate:** sono stati scelti valori in scala logaritmica a partire da 0.1 fino a 10^{-5}.
- **momentum:** Inizialmente sono stati testati i valori 0, 0.2, 0.4, 0.6, 0.8.

(a) Accuratezze medie considerando kernel_size 5

	32	**64**	**128**
50	80.07%	80.35%	80.29%
150	79.95%	80.89%	81.29%
250	80.48%	80.78%	80.75%

(b) Accuratezze medie considerando kernel_size 20

	32	**64**	**128**
50	81.30%	81.29%	81.25%
150	81.57%	81.23%	81.64%
250	81.21%	81.82%	81.85%

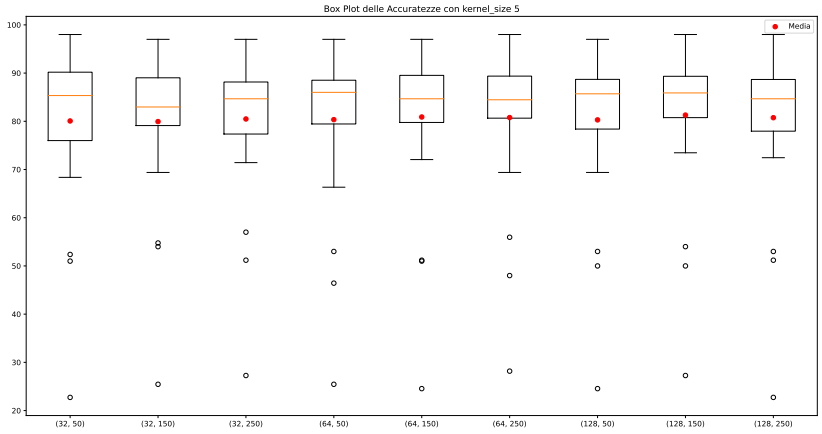


Figura 5.7: BoxPlot delle accuratezze considerando kernel_size 5

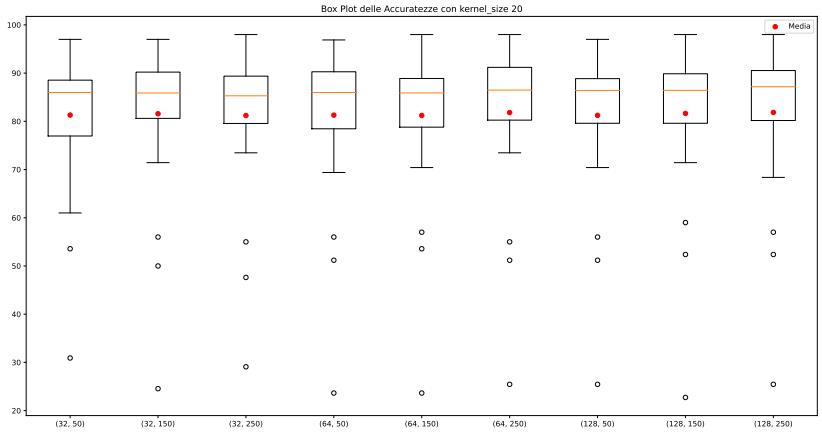


Figura 5.8: BoxPlot delle accuratezze considerando kernel_size 20

Per la fase di testing degli iperparametri learning_rate e momentun è stato scelta la migliore parametrizzazione riscontrata nei precedenti test, che corrisponde a kernel_size=20, n_filters=128 e epochs=250.

Tabella 5.5: Accuratezze medie al variare di learning_rate e momentum

	0	**0.2**	**0.4**	**0.6**	**0.8**
$\mathbf{10^{-1}}$	66.50%	51.32%	51.26%	50.05%	50.04%
$\mathbf{10^{-2}}$	81.14%	81.12%	81.79%	82.01%	81.86%
$\mathbf{10^{-3}}$	81.20%	80.27%	80.81%	81.49%	81.41%
$\mathbf{10^{-4}}$	77.69%	77.60%	78.86%	79.69%	80.13%
$\mathbf{10^{-5}}$	66.37%	66.94%	70.06%	71.81%	74.96%

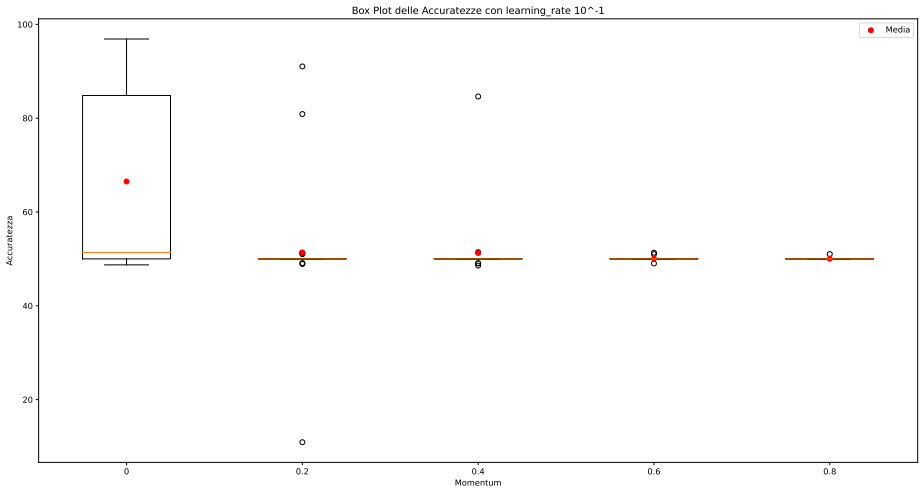


Figura 5.9: BoxPlot con learning_rate 10^{-1}

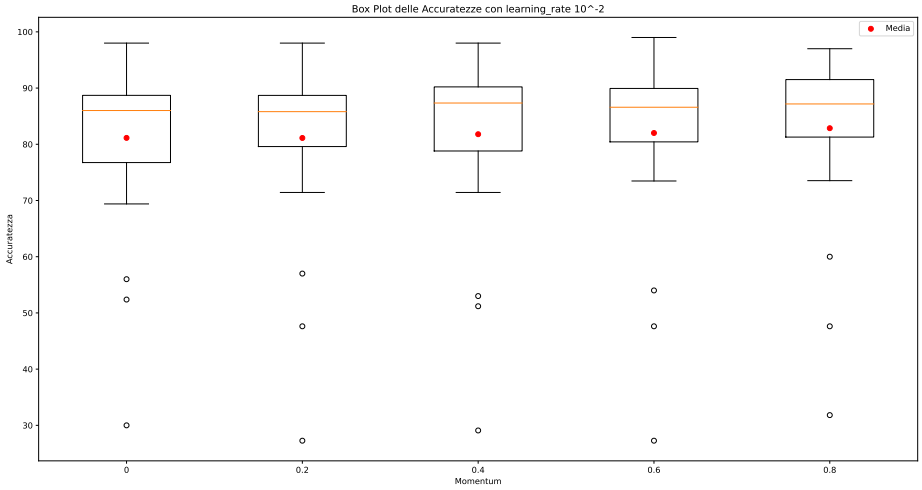


Figura 5.10: BoxPlot con learning_rate 10^{-2}

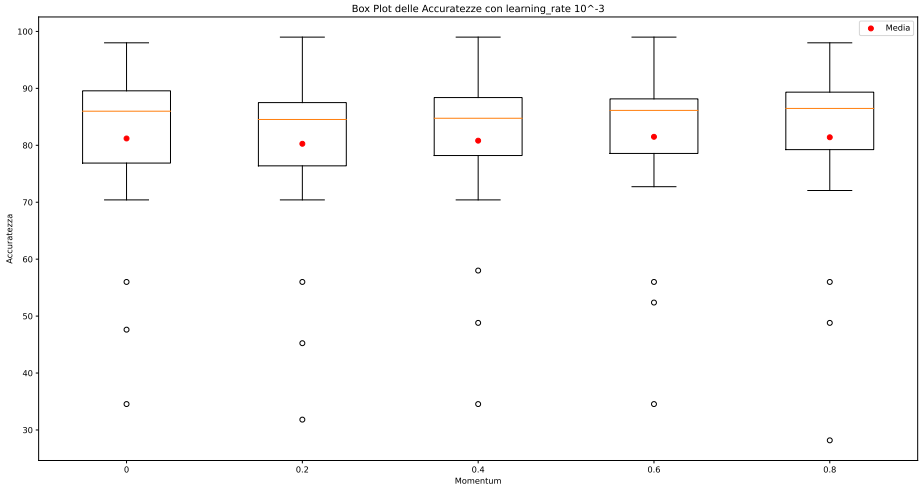


Figura 5.11: BoxPlot con learning_rate 10^{-3}

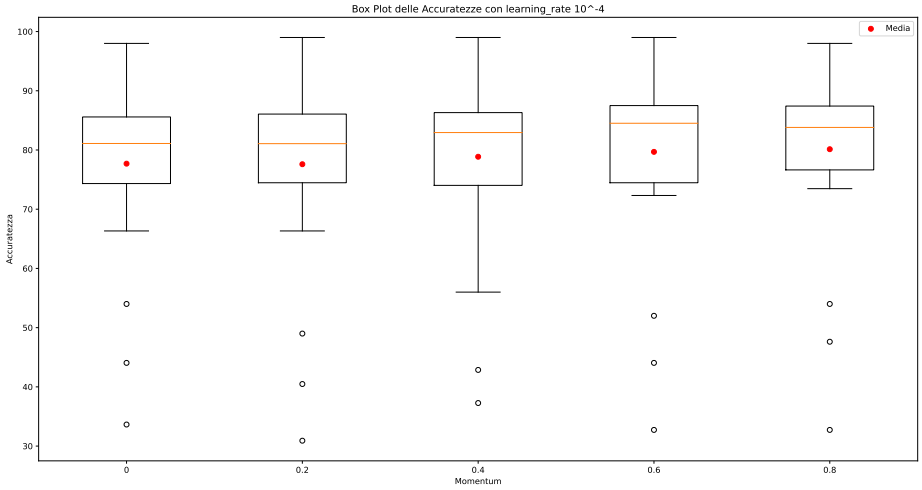


Figura 5.12: BoxPlot con learning_rate 10^{-4}

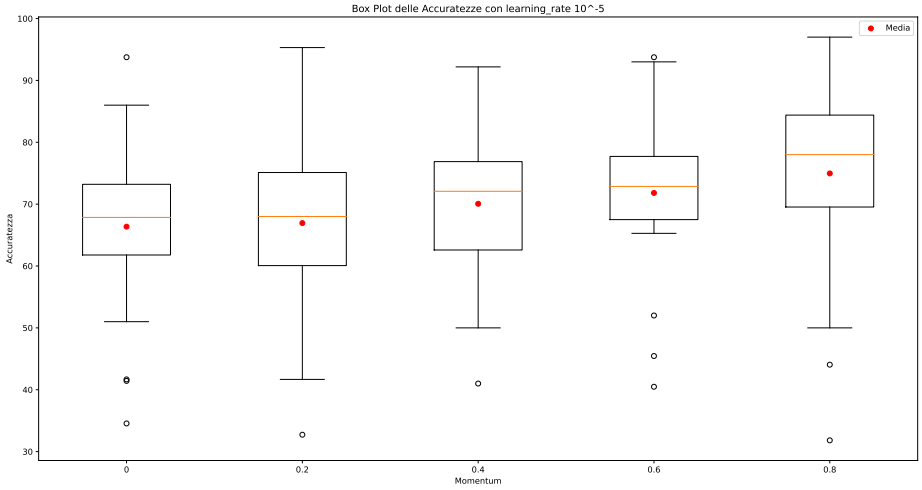


Figura 5.13: BoxPlot con learning_rate 10^{-5}

Dai risultati ottenuti possiamo ricavare alcuni dei valori migliori per il nostro modello.
Partendo dal kernel_size possiamo notare che sostanzialmente in tutti i casi studiati si ottiene una maggiore accuratezza con il valore di 20, andando poi a scegliere il valore di accuracy più alto ottenuto che corrisponde a 81,85%, che si ha con filters=128 e epochs=250. Per gli ulteriori due iperparametri, le migliori accuratezze sono ottenute per i valori di learning_rate centrali, quindi 10^{-2} e 10^{-3}. Uscendo fuori da questi valori l'accuratezza diminuisce notevolmente, notando che con un tasso di apprendimento molto alto come 0.1 il modello non converge correttamente

portando un'accuratezza del 50%. Per quanto riguarda il momentum, a parità di learning_rate, non abbiamo variazioni notevoli nei casi con maggiore accuratezza, mentre porta a variazioni più significative se abbiamo learning_rate che ci fornisce una minore accuratezza. Studiando i due valori migliori per il learning_rate notiamo come in entrambi i casi i valori massimi sono ottenuti con un momentum di 0.6 e di 0.8.

5.2 Utilizzo di una Convolutional Neural Network Bidimensionale

Dato il miglioramento delle performance ottenuto grazie al tuning degli iperparametri utilizzando una *CNN1D*, è stato scelto di provare a testare la stessa parametrizzazione trovata con un modello simile ma diverso da quello precedentemente utilizzato come la *CNN2D*.

Tabella 5.6: Statistiche sull'accuratezza della CNN2D

Max Accuracy	**Min Accuracy**	**Media Accuracy**	**Dev. Std. Accuracy**
97.00%	26.36%	80.91%	15.69%

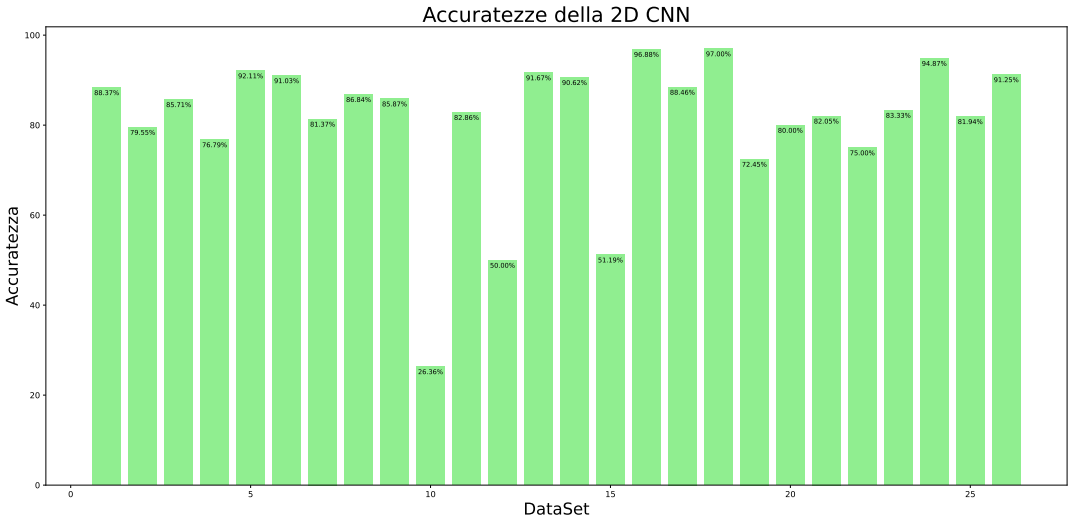


Figura 5.14: Istogramma delle accuratezze della CNN2D

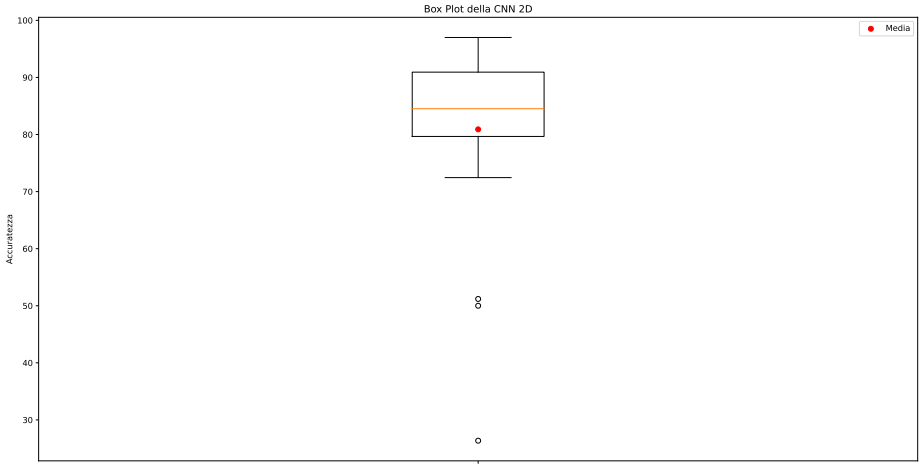


Figura 5.15: BoxPlot delle accuratezze della CNN2D

L'utilizzo di una *Convolutional Neural Network Bidimensionale* non porta ad una maggiore accuratezza rispetto all'utilizzo di una *CNN1D*, ma si ha un calo di circa un punto percentuale. Il processo di addestramento e testing di una *CNN2D* è sicuramente più pesante di quello della *CNN1D*, quindi a parità di prestazioni è preferibile scegliere il modello che computazionalmente richiede meno risorse.

5.3 Utilizzo del modello Learning Shapelets

Il modello *Learning Shapelets*, già introdotto in precedenza, fa parte della libreria tslearn, specifica per l'analisi di timeseries. Utilizzando questo modello sono stati ottenuti i seguenti risultati:

Tabella 5.7: Statistiche sull'accuratezza di Learning Shapelets

Max Accuracy	**Min Accuracy**	**Media Accuracy**	**Dev. Std. Accuracy**
58.46%	50.00%	53.46%	1.88%

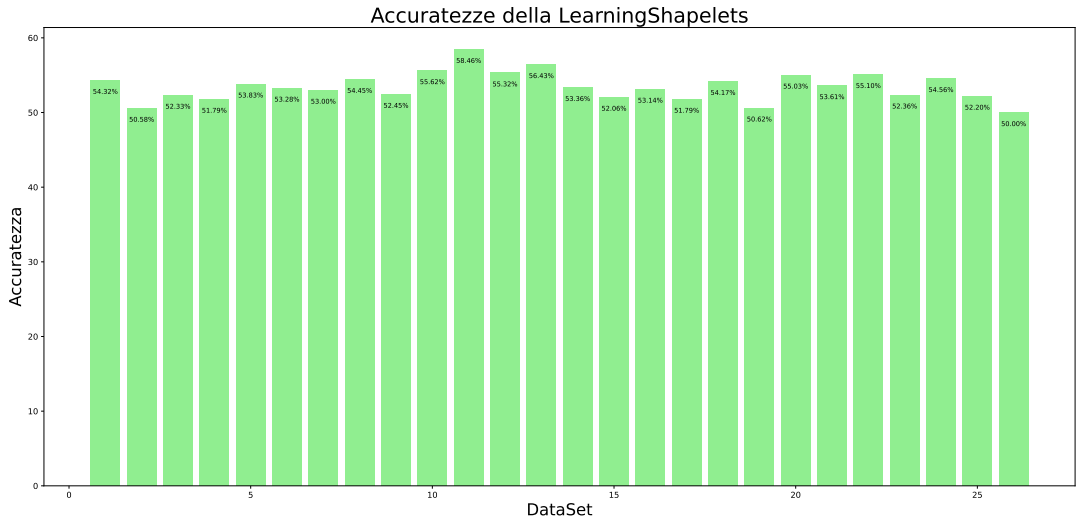


Figura 5.16: Istogramma delle accuratezze di Learning Shapelets

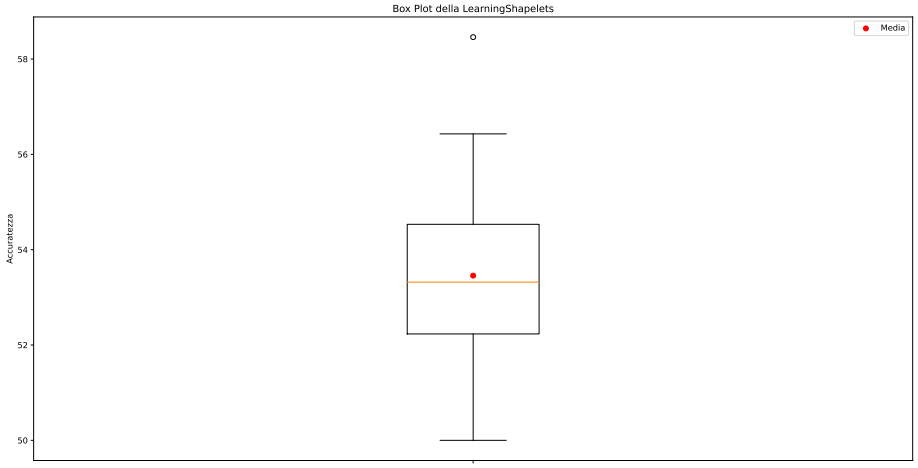


Figura 5.17: Box Plot delle accuratezze di Learning Shapelets

I risultati ottenuti con l'utilizzo di questo modello non riescono ad essere soddisfacenti a livello di performance. La accuratezza per ogni Dataset oscilla tra il 50% e il 60% con valori che sono ben distribuiti, ma che non riescono a raggiungere il livello prestazionale ottenuto utilizzando i precedenti modelli di *Convolutional Neural Networks*.

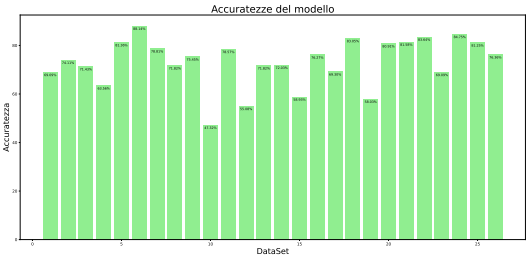
5.4 Prestazioni di una CNN1D con una diversa estrazione dei dati

La differenza principale rispetto a quanto fatto fino ad ora è la preparazione dei dati, in particolare per quanto riguarda il vettore delle etichette y. Se prima questo veniva costruito ponendo all'interno del vettore la metà delle etichette a 0 e l'altra metà delle etichette a 1, con questo nuovo metodo le etichette che costituiscono il vettore y sono poste direttamente all'interno di un file e vengono estratte da esso proprio come viene fatto con i dati.
Inoltre in questo modello vengono analizzate anche le metriche *precision, recall* e *f1_score*.

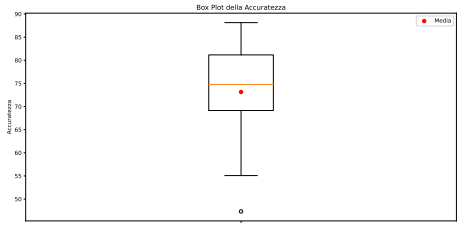
Tabella 5.8: Statistiche sull'accuratezza di Learning Shapelets

	Max	**Min**	**Media**	**Dev. Std.**
Accuratezza	88.14%	47.32%	73.14%	9.79%
Recall	83.41%	31.94%	59.94%	13.97%
Precision	82.71%	45.67%	65.79%	11.52%
F1_Score	82.66%	38.90%	61.72%	12.56%

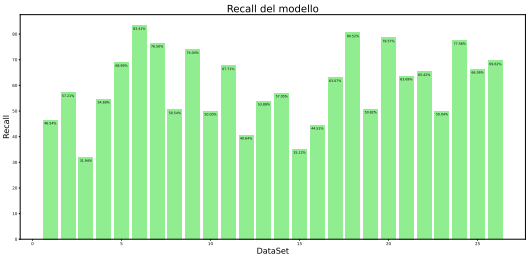
Come possiamo notare dalle analisi prestazionali del modello, questo approccio non porta ad un aumento significativo dell'accuratezza. Otteniamo una accuratezza media di circa il 73%, che è simile all'accuratezza ottenuta inizialmente. Analizzando anche le altre metriche prese in considerazione possiamo vedere che si ottiene una maggiore Precision rispetto a Recall. Questo vuol dire che il modello riesce a identificare maggiormente i casi positivi. Non si ottengono comunque delle percentuali elevate per entrambe le metriche, il che ci suggerisce che il modello non riesce a identificare correttamente un grande numero di True Positives. Ottenendo queste due metriche non elevate si ha che di conseguenza anche la metrica F1_Score non avrà valori elevati, avendo una media di circa il 62%.



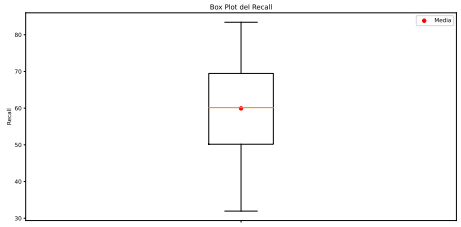
(a) Istogramma della Accuratezza



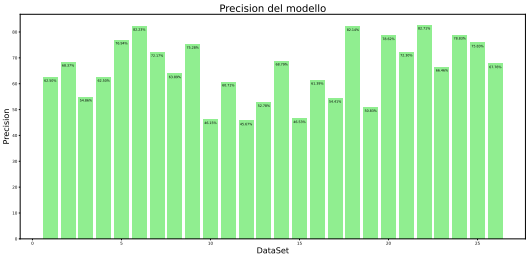
(b) Box Plot della Accuratezza



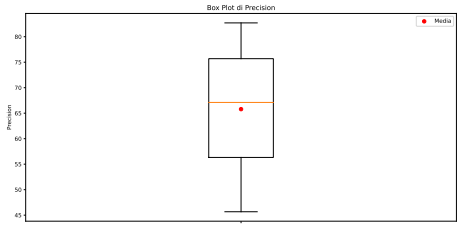
(c) Istogramma di Recall



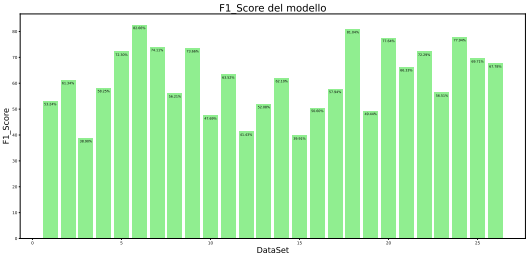
(d) Box Plot di Recall



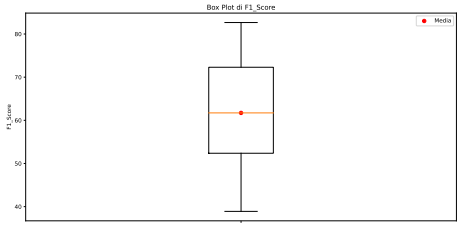
(e) Istogramma di Precision



(f) Box Plot di Precision



(g) Istogramma di F1_Score



(h) Box Plot di F1_Score

Figura 5.18: Istogrammi e Box Plot

Capitolo 6

Conclusioni

6.1 Considerazioni finali

Partendo dai risultati sperimentali ottenuti ed analizzati è possibile trarre alcune fondamentali conclusioni:

- **Non tutti i modelli possono essere utilizzati per lo stesso dataset:** in questo studio sono stati utilizzati tre tipi di modelli, *CNN1D, CNN2D e Learning Shapelets*, ma non tutti hanno restituito le stesse prestazioni sugli stessi dataset. Difatti, se i due tipi di *Convolutional Neural Network* hanno restituito performance simili, lo stesso non può essere affermato per il terzo modello, che ci ha restituito prestazioni decisamente inferiori.
  Questo conferma quanto già detto in apertura, ovvero che ogni modello può essere utile se utilizzato su dataset che possano esaltare le sue caratteristiche, altrimenti utilizzando su un dataset il modello sbagliato non verranno mai raggiunte le prestazioni desiderate.

- **L'ottimizzazione del modello è fondamentale per l'incremento delle prestazioni:**
  Come affermato inizalmente, il tuning degli iperparametri e la sperimentazione di alcuni possibili combinazioni all'interno di un modello è un processo molto importante per studiare il comportamento del modello sul dataset e per migliorare di conseguenza le sue prestazioni. I risultati sperimentali otteuti riguardanti l'ottimizzazione del modello danno la conferma di quanto affermato. Se in un primo momento l'accuratezza del modello si aggirava intorno al 74%, grazie all'esplorazione di varie possibilità di combinare iperparametri, ottimizzatori e metodi di pooling è stato possibile incrementare l'accuratezza del modello di circa 8 punti percentuali, arrivando ad ottenere più dell'82% grazie a semplici operazioni di ottimizzazione.

La miglior prestazione sperimentata è stata ottenuta utilizzando una *Convolutional Neural Network Unidimensionale* con la seguente configurazione:

- **batch_size** posto a 8.
- **kernel_size** posto a 5.
- **pool_size** posto a 2.
- **filters** posto a 64.
- **epochs** posto a 150.
- **Metodo di pooling** MaxPooling.
- **Ottimizzatore** Adam con parametri di default.

Grazie a questa parametrizzazione sperimentata è stato possibile ottenere una accuratezza di 82.25% (Tabella 5.3), incrementando notevolmente le prestazioni del nostro modello come già affermato.

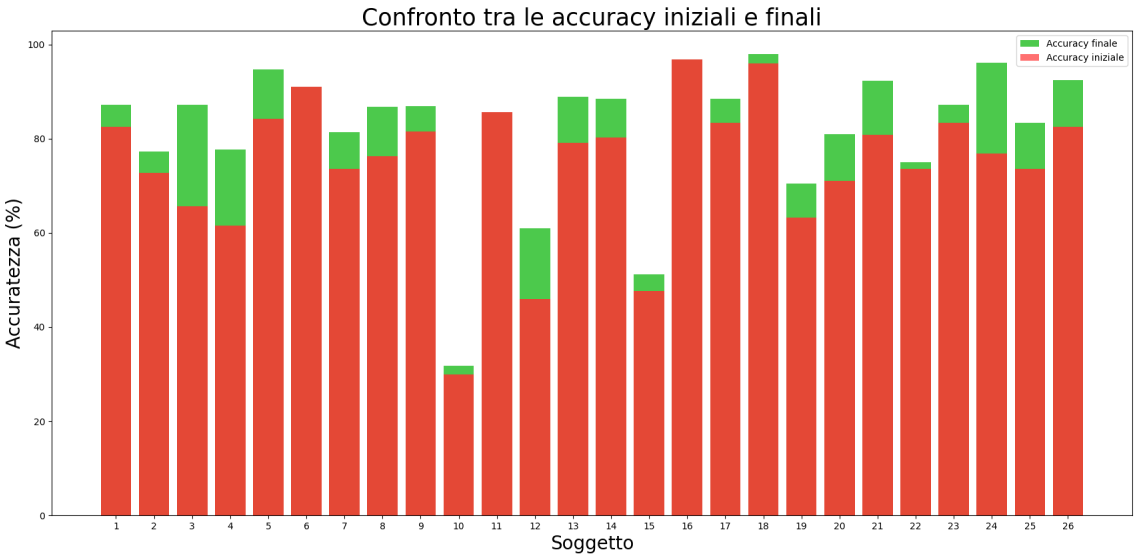


Figura 6.1: Confronto tra le accuracy iniziali e finali

È doverso sottolinerare che anche altre parametrizzazioni hanno ottenuto risultati simili a questo come ad esempio la seguente sempre relativa ad una CNN1D:

- **batch_size** posto a 8.
- **kernel_size** posto a 20.
- **pool_size** posto a 2.
- **filters** posto a 128.
- **epochs** posto a 250.

- **Metodo di pooling** MaxPooling.
- **Ottimizzatore** SGD.
- **learning_rate** posto a 0.01.
- **momentum** posto a 0.6.

Con questa configurazione è stato possibile una accuratezza di 82.01%(Tabella 5.5), molto vicina alla prestazione massima raggiunta.
Sostanzialmente è possibile affermare che in termini di performance le due parametrizzazioni si equivalgono, questo perchè i risultati ottenuti sono sicuramente influenzati dalla situazionalità del singolo test effettuato.
Due esecuzioni consecutive dell'addestramento e il testing di uno stesso modello con la stessa parametrizzazione è difficile che restituiscano la stessa accuratezza. Eseguendo più volte l'addestramento e il testing di un modello identico, e facendo la media delle prestazioni ottenute in ogni esecuzione, è possibile avere una stima più precisa delle performance del modello e ridurre al minimo possibile l'aleatorietà delle singole esecuzioni.

6.2 Applicazioni e sviluppi futuri

Come già detto in precendenza lo sviluppo di modelli di *Machine Learning* per l'analisi di segnali fisiologici trova possibili applicazioni in molti campi, a partire sopratutto da quello medico.
L'integrazione dell'*Intelligenza Artificiale* nella vita di tutti i giorni è necessaria per riuscire a svolgere le mansioni con più semplicità e con maggiore sicurezza.
I risultati ottenuti in questo studio permettono l'utilizzo di un modello per la classificazione di segnali fisiologici con prestazioni che possono essere definite buone, sopratutto analizzando il percorso di crescita effettuato da esse, anche se non eccelse, ma che sono sicuramente migliorabili.
Per il miglioramento delle performance possono essere intraprese varie strade, dall'esplorazione di ulteriori parametrizzazioni non ancora sperimentate, la modifica della struttura del modello, come l'aggiunta di più livelli o più neuroni, oppure sperimentare modelli differenti che potrebbero adattarsi meglio a questo tipo di dataset. Oltre al miglioramento dell'accuratezza è possibile anche sviluppare future ricerche sulla riduzione del costo computazionale, unica vera criticità emersa nel corso di questo studio, dato che a parità di prestazioni è sempre preferibile scegliere un modello o una parametrizzazione che permette un uso efficente e moderato delle risorse.

Bibliografia

[1] Diego Candia-Rivera, Vincenzo Catrambone, Riccardo Barbieri, and Gaetano Valenza. Functional assessment of bidirectional cortical and peripheral neural control on heartbeat dynamics: A brain-heart study on thermal stress. 2022.

[2] Adman Ghaderi, Javad Frounchi, and Alireza Farnam. Machine learning-based signal processing using physiological signals for stress detection. 2015.

[3] L. Gonzalez-Carabarin, E.A. Castellanos Alvarado, P. Castro-Garcia, and Garcia-Ramirez M.A. Machine learning for personalised stress detection: Inter-individual variability of eeg-ecg markers for acute-stress response. 2021.

[4] Haider Khalaf Jabbar and Rafiqul Zaman Khan. Methods to avoid over-fitting and under-fitting in supervised machine learning. 2015.

[5] Awais Mehmood, Munwar Iqbal, Zahid Mehmood, Aun Irtaza, Marrian Nawaz, Tahira Nazir, and Momina Masood. Prediction of heart disease using deep convolutional neural networks. 46:3409–3422, 2021.

[6] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. 2015.

[7] Vasco Ponciano, Ivan Miguel Pires, Fernando Reinaldo Ribeiro, Nuno M. Garcia, Maria Vanessa Villasana, Eftim Zdravevski, and Petre Lameski. Machine learning techniques with ecg and eeg data: An exploratory study5. 2020.

[8] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. 2019.

[9] Kei Suzuki, Tipporn Laohakangvalvit, Ryota Matsubara, and Midori Sugaya. Constructing an emotion estimation model based on eeg/hrv indexes using feature extraction and feature selection algorithms. 2021.

[10] Hilde J.P. Weerts, Andreas C. Müller, and Joaquin Vanschoren. Importance of tuning hyperparameters of machine learning algorithms. 2020.

[11] Mikolaj Wojciuk, Zaneta Swiderska-Chadaj, Krzysztf Siwek, and Arkadiusz Gertych. The role of hyperparameter optimization in fine-tuning of cnn models. 2022.

[12] David Zambrana-Vinaroz, José María Vicente-Samper, Juliana Manrique-Cordoba, and Sabater-Navarro José María. Wearable epileptic seizure prediction system based on machine learning techniques using ecg, ppg and eeg signals. 2022.