Visual ECG Analysis

Gaetano Cimino
Emilio Mainardi
Giulio Varone
Dipartimento di Informatica
Università degli Studi di Salerno
{g.cimino2, e.mainardi, g.varone7}@studenti.unisa.it

***Abstract* – L'elettrocardiogramma (ECG) può essere utilizzato in modo affidabile come misura per monitorare la funzionalità del sistema cardiovascolare. Di recente, c'è stata una grande attenzione verso un'accurata categorizzazione dei battiti cardiaci. Il rilevamento precoce e accurato dei tipi di aritmia è importante per rilevare le malattie cardiache e scegliere un trattamento adeguato per il paziente. Tra tutti i classificatori, le reti neurali convoluzionali (CNN) sono diventate molto popolari per la classificazione ECG. Questo articolo discute le problematiche relative alla classificazione dell'ECG e presenta un'indagine dettagliata di tecniche di pre-processing su dataset di ECG, tecniche di feature extraction e classificatori basati su CNN. In particolare, la novità introdotta rispetto ad altri metodi proposti in letteratura è quella di effettuare una classificazione di ECG su fonti video ed in real-time. Secondo i risultati, il metodo suggerito è in grado di fare previsioni con accuratezza media del 98,47%.**

I. INTRODUZIONE

L'elettrocardiogramma (ECG) viene utilizzato abitualmente nella pratica clinica e descrive l'attività elettrica del cuore. Nei controlli ospedalieri, i medici registrano l'ECG per monitorare le condizioni cardiache dei pazienti. I medici interpretano le forme delle onde e calcolano i parametri per determinare se l'ECG mostra segni di malattia cardiaca o meno. Queste operazioni sono una routine noiosa per il medico. Pertanto, è necessario un sistema di riconoscimento automatico dell'ECG per ridurre l'onere dell'interpretazione dello stesso ed avere una diagnosi accurata e in tempi brevi dei battiti cardiaci aritmici [1].

Per affrontare i problemi sollevati con l'analisi manuale dei segnali ECG, molti studi in letteratura utilizzano tecniche di machine learning per rilevare accuratamente le anomalie nel segnale. Sono stati condotti vari studi per la classificazione di diverse aritmie cardiache. Attualmente, le classificazioni di ECG vengono effettuate solo su singoli segnali estratti da immagini.

In questo articolo, proponiamo un'estensione dei metodi di classificazione di ECG tramite l'utilizzo di reti neurali convoluzionali (CNN) effettuando tale operazione su fonti video ed in real-time, in modo tale che un medico possa fornire un responso in tempi più brevi. Inoltre, effettuare una classificazione in real-time permette di individuare le anomalie dei segnali nell'immediato ed intervenire prontamente in situazioni di emergenza. Viene utilizzato un metodo basato sul modello HSB (Hue Saturation Brightness) per effettuare un processo di feature extraction attuo a ricostruire un segnale; viene poi utilizzata una CNN per classificare il battito cardiaco dell'ECG. I dataset del mondo reale, tuttavia, sono caratterizzati da una quantità elevata di feature che richiedono un tempo estremamente elevato per effettuare il training dei modelli. Pertanto, sono state adottate delle tecniche per effettuare un processo di dimensionality reduction, attraverso l'utilizzo di Dipendenze Funzionali (FD).

L'approccio proposto è validato sul dataset dell'aritmia del MIT-BIH e ottiene un'elevata precisione di classificazione.

II. STATO DELL'ARTE

In letteratura ci sono vari metodi per la classificazione di singoli segnali ECG.

Il metodo proposto da Kachuee, Fazeli e Sarrafzadeh dell'Università della California [2] utilizza una deep neural network per la classificazione del battito cardiaco in diverse categorie, partendo dai casi normali e analizzando le diverse aritmie. L'approccio inizialmente prevede una fase di preprocessing dove viene effettuata una normalizzazione dei valori di ampiezza del segnale ECG continuo, e viene rilevato ciascun picco R, selezionando successivamente una parte del segnale con lunghezza 1:2T, dove T indica un tempo prefissato. Per la fase di classificazione viene utilizzata una CNN sul dataset MIT-BIH suddiviso per l'80% per il training set e il 20% per il test set.

Il metodo proposto da Zhao e Zhang dell'Università di Shangai [3] comprende tre fasi tra cui la preelaborazione dei dati, l'estrazione delle caratteristiche e la classificazione dei segnali ECG. Viene utilizzata la trasformata di Wavelet per estrarre i coefficienti della trasformata come caratteristiche di ciascun segmento ECG. Infine, viene proposto l'utilizzo della Support Vector Machine (SVM) con kernel gaussiano per classificare il ritmo cardiaco ECG.

Il metodo proposto da Shi, Wang, Huang, Zhao, Qin, Liu dell'Università di Shangai [4] comprende un metodo di classificazione gerarchico basato sul weighted extreme gradient boosting (XGBoost). Un gran numero di features di 6 categorie diverse (es. statistiche,intervallari,etc.) vengono estratte dai battiti cardiaci pre-elaborati tramite la trasformata di Wavelet. Viene quindi utilizzata l'eliminazione ricorsiva delle caratteristiche (RFE) per selezionare quelle rilevanti. Successivamente, un classificatore gerarchico è costruito in fase di training e viene migliorato aggiornando i pesi.

Il metodo proposto da Alarsan e Younes [5] viene valutato e convalidato sul dataset dell'aritmia MIT-BIH. Viene proposto un metodo di classificazione basato su Gradient Boosted Trees, Decision Tree e Random Forest. Per la feature extraction viene utilizzata la trasformata discreta di Wavelet.

Il metodo proposto da Zhou [6] propone una architettura composta da ResNet + Bi-LSTM, che combina i vantaggi di entrambi per estrarre caratteristiche locali rilevanti e per effettuare la classificazione dei segnali ECG.

Tutti i lavori citati si limitano ad effettuare classificazioni di singoli segnali ECG estratti da immagini.

III. APPROCCIO

L'approccio proposto è quello di effettuare il training di una rete neurale convoluzionale (CNN) su dei segnali ECG recuperati dal dataset MIT-BIH.

Prima di procedere con il training del modello è stata effettuata una fase di Data Preparation sul dataset.

Dalle analisi è stato rilevato uno sbilanciamento tra le classi, pertanto si è ritenuto necessario effettuare un processo di resampling. Inoltre, siccome il numero di feature di cui il dataset è caratterizzato è molto elevato, è stato necessario effettuare un processo di dimensionality reduction, tramite l'utilizzo di Dipendenze Funzionali (FD).

Infine, in tale sezione viene illustrata l'architettura realizzata per la CNN.

A. Dataset MIT-BIH

Il dataset dell'aritmia del MIT-BIH [7] è rappresentato da 48 estratti di mezz'ora di registrazioni ECG ambulatoriali registrati alla frequenza di campionamento di 360Hz, ottenute da 47 soggetti studiati dal laboratorio di aritmia della BIH tra il 1975 e il 1979. Ogni battito è stato annotato da almeno due cardiologi. Le posizioni delle annotazioni dei battiti sono state regolate attraverso un software che filtra digitalmente il segnale primario (MLII); le posizioni delle annotazioni dei battiti corrotti dal rumore sono state riposizionate manualmente. Circa la metà delle registrazioni sono state scelte a caso da un set di 4000 ECG; le restanti sono state selezionate dallo stesso set per includere aritmie meno comuni ma clinicamente significative che non sarebbero ben rappresentate in un piccolo campione casuale.

Il successo del dataset MIT-BIH ha dimostrato il valore della creazione di collezioni di ECG generalmente disponibili, rappresentative e ben

caratterizzate. Le esigenze dei ricercatori accademici si sta spostando verso un'analisi più accurata e dettagliata di ECG di un singolo soggetto, considerando un periodo temporale di 24 o 48 ore.

Il dataset è composto da 87554 istanze, dove ciascuna rappresenta un segnale ECG. Ciascuna istanza è composta da 188 feature (dove ognuna viene specificata attraverso un numero univoco che la identifica), tra cui l'ultima rappresenta la label di un singolo segnale, cioè la classe associata ad esso, che viene espressa attraverso un intero da 0 a 4.

La *Tabella 3.2* mostra le diverse classi da poter associare ad un segnale ECG [2]. Le classi sono state definite in conformità con lo standard EC57 dell'Association for the Advancement of Medical Instrumentation (AAMI).

Tutte le altre feature sono dei float normalizzati nel range (0,1) che rappresentano i valori delle coordinate Y che compongono il vettore delle caratteristiche estratto per un determinato segnale.

La *Tabella 3.1* mostra un estratto del dataset.

	0	1	2	…	186	187
0	0,977	0,926	0,681	…	0	0
1	0,960	0,863	0,461	…	0	0
2	1	0,659	0,186	…	0	0
3	0,925	0,665	0,541	…	0	0
4	0,967	1	0,830	…	0	0
…	…	…	…	…	…	…
87553	0,901	0,845	0,800	…	0	4

Tabella 3.1: estratto del dataset

Categoria	Annotazioni
N	• Normal • Left/Right bundle branch block • Atrial escape • Nodal escape
S	• Atrial premature • Aberrant atrial premature • Nodal premature • Supra-ventricular premature
V	• Premature ventricular contraction • Ventricular escape
F	• Fusion of ventricular and normal
U	• Paced • Fusion of paced and normal • Unclassifiable

Tabella 3.2: tipologie di classi per segnali ECG

La categoria '**N**' viene assegnata ad un ECG che presenta un battito regolare (tra i 60 e i 100 bpm).

La categoria '**S**' viene assegnata ad un ECG che presenta un'aritmia con un battito in più determinata da un'attivazione elettrica anomala. Questi battiti sono comuni nei soggetti in età avanzata.

La categoria '**V**' viene assegnata ad un ECG che presenta battiti prematuri ventricolari, i quali sono singoli impulsi ventricolari dovuti a un rientro nel ventricolo o a un anomalo automatismo delle cellule ventricolari. Possono essere avvertiti come battiti mancati o saltati e sono più frequenti con gli stimolanti (ansia, stress, alcool, caffeina).

La categoria '**F**' viene assegnata ad un ECG che presenta una combinazione di caratteristiche dei segnali ECG di classe N e di classe V.

La categoria '**U**' viene assegnata ad un ECG per cui non viene riconosciuta alcuna categoria precedente, quindi viene classificato come battito sconosciuto.

B. Resampling del dataset MIT-BIH

Innanzitutto, analizzando il dataset è stato rilevato uno sbilanciamento dei campioni nelle diverse classi. Il problema derivante da ciò è che i modelli addestrati su set di dati non bilanciati hanno spesso scarsi risultati quando devono generalizzare su nuove istanze.

Tale fenomeno viene evidenziato nella *Figura 3.1*, dove viene mostrata la distribuzione delle frequenze delle varie classi attraverso un bar-plot. In particolare, sull'asse delle ordinate viene illustrata la quantità di istanze in relazione ad una determinata classe, mentre sull'asse delle ascisse vengono rappresentate le varie classi.

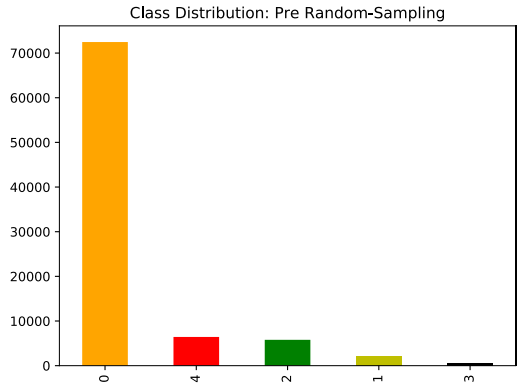

Figura 3.1: bar-plot della distribuzione delle classi pre - resampling

Pertanto, è stato necessario effettuare il re-sampling del dataset, utilizzando sia la tecnica dell'*oversampling* che dell'*undersampling*[8]. Le motivazioni di tale scelta vengono illustrate nella sezione **VI.A**.

L'oversampling è un metodo non euristico che mira a bilanciare la distribuzione di classe attraverso la replicazione di esempi di classe minoritari. I meccanismi di oversampling prevedono l'aggiunta di un insieme E ottenuto dalle osservazioni appartenenti alle classi di minoranza. In questo modo, il numero di osservazioni totali è aumentato e la distribuzione della classe viene equilibrata. Questo fornisce un meccanismo per variare il grado di equilibrio di una distribuzione di classi sbilanciate a qualsiasi livello desiderato.

L'undersampling è un metodo non euristico che mira a bilanciare la distribuzione di classe attraverso l'eliminazione di osservazioni appartenenti alla classe di maggioranza. Poiché molti esempi della classe di maggioranza vengono eliminati, il training set diventa più equilibrato e il processo di apprendimento diventa più veloce.

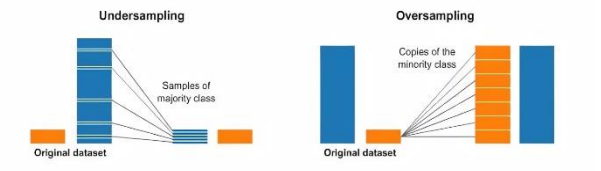

Figura 3.2: oversampling e undersampling

[8] https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

Per quanto riguarda le istanze della classe 'N' è stato necessario applicare una tecnica di undersampling, in quanto il numero di istanze presenti all'interno del dataset per tale classe risulta prevalere sul numero di istanze delle altre classi. In particolare, per tale classe, è stato utilizzato il metodo *sample* che campiona in modo casuale un sottoinsieme di lunghezza prefissata di elementi scelti dalla sequenza data in input. Mentre, per le altre classi, è stato necessario applicare una tecnica di oversampling, in quanto il numero di istanze presenti all'interno del dataset per tali classi risulta essere molto basso. Pertanto, è stato utilizzato il metodo *resample* che, a partire dalle istanze presenti per una classe, genera un numero prefissato di istanze simili.

La *Figura 3.3* mostra la distribuzione delle frequenze delle varie classi dopo aver effettuato il resampling del dataset, sempre attraverso l'utilizzo di un bar-plot.

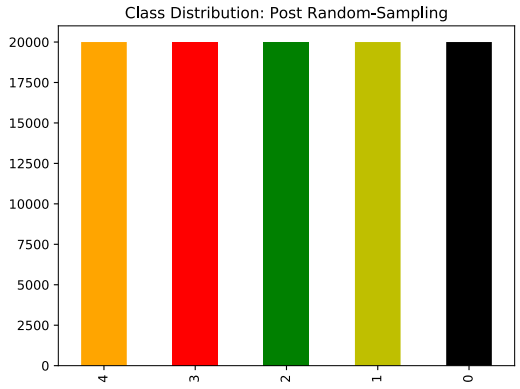

Figura 3.3: bar-plot della distribuzione delle classi post - resampling

Come si può notare, per ogni classe sono state prese in considerazione 20.000 istanze (settando il parametro *n* sia del metodo sample che resample a tale valore), considerando quindi un totale di 100.000 istanze. Infine, è stato settato il parametro *random_state* per ottenere gli stessi risultati su diverse chiamate di funzione.

Nella fase successiva sono state prese in considerazione le Dipendenze Funzionali per attuare il processo di *dimensionality reduction* precedentemente accennato.

C. *Feature Selection tramite FD*

La *Feature Selection* è un processo in cui si selezionano automaticamente o manualmente le feature che contribuiscono maggiormente al valore di previsione da associare alla variabile target a cui si è interessati. I vantaggi di eseguire un processo di Feature Selection prima di modellare i dati sono:

- Ridurre la probabilità di avere overfitting: avere dati meno ridondanti significa ridurre l'opportunità di prendere decisioni basate su rumore;
- Ridurre i tempi di addestramento: un numero inferiore di feature riduce la complessità degli algoritmi e quest'ultimi si allenano più rapidamente.

In particolare, feature irrilevanti o parzialmente rilevanti, possono influire negativamente sulle performance di un modello. Quindi, processi di Feature Selection e di Data Cleaning dovrebbero essere il primo e il più importante passo nella progettazione di un modello.

Esistono vari modi per effettuare un processo di Feature Selection. In questo progetto si è deciso di utilizzare particolari metadati, cioè le *Dipendenze Funzionali*.

Una Dipendenza Funzionale tra due insiemi di attributi X e Y di un dataset specifica un vincolo sulle tuple che possono formare uno stato di relazione del dataset. Indicato con $X \rightarrow Y$, il vincolo è che per ogni coppia di tuple t_1 e t_2, per le quali vale che $t_1[X] = t_2[X]$, allora deve valere anche $t_1[Y] = t_2[Y]$.

La motivazione di tale scelta è che una Dipendenza Funzionale descrive un vincolo semantico tra due insiemi di attributi di un dataset. Questo ci permette di individuare una correlazione tra attributi; tale correlazione tra due insiemi di attributi rende inutile l'utilizzo di entrambi. Questo ci consente di utilizzare pochi attributi che sintetizzano i restanti. Pertanto, utilizzando un sottoinsieme di attributi individuato in questo modo, si preserva la maggior parte delle informazioni ma, allo stesso tempo, si riduce la dimensionalità di un dataset.

Per poter effettuare l'estrazione di FD valide su un dataset si utilizza un algoritmo di discovery. Esistono varie tipologie di algoritmi di discovery di FD, nel dettaglio abbiamo quelli row-based, column-based e quelli ibridi (che utilizzano una combinazione dei due precedenti approcci). Quelli row-based effettuano l'individuazione andando a considerare combinazioni di tuple che caratterizzano un dataset, mentre quelli column-based effettuano l'individuazione andando ad analizzare combinazioni di colonne del dataset e si basano sull'esplorazione di una struttura a lattice che viene utilizzata per effettuare la rappresentazione dello spazio di ricerca.

Ovviamente, se viene effettuato il discovery di FD tramite l'utilizzo di un algoritmo column-based, più è grande il numero di feature di cui un dataset è caratterizzato, più cresce il tempo necessario per poter estrarre delle FD, in quanto aumenta lo spazio di ricerca. Siccome utilizziamo un algoritmo column-based per effettuare il discovery su un dataset con oltre 180 feature, tale processo potrebbe impiegare molto tempo e molta memoria. Pertanto, per poter effettuare l'estrazione delle dipendenze è stato necessario effettuare un partizionamento verticale del dataset, cioè è stato partizionato per colonne, con l'obiettivo di individuare il maggior numero di dipendenze valide sul dataset. Un frammento verticale del dataset mantiene solo determinati attributi di esso. Tramite l'utilizzo dell'algoritmo di discovery COD3 sono state estratte delle FD minimali per ogni partizionamento. La scelta del partizionamento verticale è dovuta al fatto che COD3 è un algoritmo column-based e pertanto è molto veloce su dataset con un numero di colonne relativamente basso. Inoltre, trattandosi di FD canoniche, il partizionamento verticale garantisce la correttezza del risultato, in quanto ogni singola FD deve valere sull'intero set di dati, cioè su ogni tupla secondo la definizione.

Parliamo di FD minimale quando, a partire da una FD individuata sul dataset, rimuovendo anche solo un attributo dal lato sinistro di essa, non risulta essere più valida. In particolare, tra le varie FD minimali estratte sul dataset, ci siamo focalizzati solo su quelle caratterizzate dal minor numero di attributi sul lato sinistro. Pertanto, sono state considerate 16 FD minimali; ovviamente quest'ultime non rappresentano l'insieme completo di FD estratte sul dataset, ma sono sicuramente corrette e minimali per le proprietà dell'algoritmo utilizzato.

Considerando quindi le FD appena citate, con pochi attributi riusciamo a minimizzare il numero di feature da utilizzare, questo perché una FD esprime una relazione di stretta uguaglianza tra gli attributi, dunque possiamo non utilizzare, sia per la fase di training che per il processo di Feature Extraction, gli attributi che si trovano sul lato destro delle FD, poiché risultano essere le feature meno rilevanti per effettuare le previsioni. Ovviamente, bisogna sempre considerare il fatto che la dimensionality reduction ci fa perdere alcune informazioni: quindi, da un lato ci consente di accelerare il processo di training, però questo va a discapito delle performance. Cerchiamo di raggiungere il giusto compromesso tra velocità della fase di addestramento e qualità dei risultati. Usare tali tecniche rende la pipeline di training più complessa, questo perché c'è bisogno di addestrare prima il modello sui dati originali e, solo se la fase di addestramento è lenta, allora bisogna ricorrere alla dimensionality reduction.

Il numero totale di FD individuate dai 16 partizionamenti presi in considerazione è 132.

Utilizzando la seguente regola di inferenza:

$$\{X \to Y, X \to Z\} \vDash X \to YZ$$

Equazione 3.1: (Union (or additive) rule)

possiamo concludere che le FD selezionate sono quelle rappresentate nella *Tabella 3.3*.

(2 3 7) → (1 4 5 6 8 9 10)
(43 44 50 52) → (45 46 47 48 49 51 53)
(32 34 40) → (33 35 36 37 38 39 41 42)
(65 68 70 73) → (66 67 69 71 72 74 75)
(54 55 61 63) → (56 57 58 59 60 62 64)
(76 79 81 84) → (77 78 80 82 83 85 86)
(98 99 100 101) → (102 103 104 105 106 107 108)
(87 88 90 92) → (89 91 93 94 95 96 97)
(11 12 13 14) → (15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31)
(109 110 114) → (111 112 113 115 116 117 118 119)
(120 121 122 124) → (123 125 126 127 128 129 130)
(131 132 137) → (133 134 135 136 138 139 140 141)
(142 143 144) → (145 146 147 148 149 150 151 152)
(153 154 158) → (155 156 157 159 160 161 162 163)
(164 166) → (165 167 168 169 170 171 172 173 174)
(175 180) → (176 177 178 179 181 182 183 184 185 186)

Tabella 3.3: FD selezionate

Inizialmente per poter selezionare la combinazione di colonne del dataset da utilizzare si è pensato di utilizzare un approccio top-down, cioè partire col dataset caratterizzato da tutte le feature e man mano rimuovere un sottoinsieme di quelle individuate tramite le FD ed analizzare le varie soluzioni ottenute. Si è notato però che procedendo in questo modo il numero di combinazioni di colonne da dover analizzare fosse troppo elevato. Pertanto, si è deciso di procedere nel modo inverso tramite un approccio bottom-up, cioè partire dal dataset con tutte le feature meno rilevanti rimosse e andandole a riaggiungere man mano.

Sono state dunque rimosse dal dataset tutte le feature meno rilevanti individuate tramite le FD. Le motivazioni di tale scelta vengono illustrate nella sezione **VI.B**.

Dunque, le feature meno rilevanti risultanti sono:

- 1,4,5,6,8,9,10
- 15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31
- 33,35,36,37,38,39,41,42
- 45,46,47,48,49,51,53
- 56,57,58,59,60,62,64
- 66,67,69,71,72,74,75
- 77,78,80,82,83,85,86
- 89,91,93,94,95,96,97
- 102,103,104,105,106,107,108
- 111,112,113,115,116,117,118,119
- 123,125,126,127,128,129,130
- 133,134,135,136,138,139,140,141
- 145,146,147,148,149,150,151,152
- 155,156,157,159,160,161,162,163

o 165,167,168,169,170,171,172,173,174

o 176,177,178,179,181,182,183,184,185,186

Come si può notare, è stato possibile ridurre il numero di feature del dataset da 187 a 55 (senza considerare la label).

D. Rete Neurale Convoluzionale (CNN)

Le reti neurali convoluzionali (CNN) sono una categoria di reti neurali che si sono dimostrati molto efficaci in settori quali il riconoscimento e la classificazione delle immagini.

L'idea dietro una CNN è applicare un filtro alle immagini prima che queste vengano processate da una Deep Neural Network. La CNN è formata da uno strato di input, uno strato di output e da una serie di layer che insieme costituiscono il blocco base che è solitamente presente in ogni CNN e sono posti tra loro in cascata[9]. Di seguito, si riportano le principali caratteristiche di una CNN.

Input Layer: il primo layer è quello che riceve l'immagine e la ridimensiona prima di passarla ai layer successivi.

Convolutional Layer: lo scopo dello strato di convoluzione è quello di estrarre le feature, cioè le caratteristiche significative delle immagini che poi verranno utilizzate per calcolare i match tra i punti caratteristici durante il testing. Quindi, permettono di mappare un'intera regione dell'immagine a un singolo neurone; i neuroni del primo livello convoluzionale sono connessi direttamente ai pixel dei loro campi ricettivi. Il risultato fornito da uno strato di questo tipo dipende dal numero di filtri usati, dallo stride, ovvero lo spazio che andiamo ad introdurre che ci consente di ridurre la dimensione di uno strato andando allo strato successivo e, infine, dallo zero-padding, cioè il numero di zero di padding usati per completare la matrice di input lungo i bordi.

Rectified Linear Unit (ReLU): è una funzione di attivazione utilizzata dopo l'operazione di convoluzione. Essa è definita come:

$$f(x) = x^{+} = \max(0, x)$$

Equazione 3.2: funzione di attivazione ReLU

Con la ReLU viene introdotta la non linearità nella rete, dal momento che ogni dato reale che sarà usato sarà non lineare, al contrario della convoluzione. Questo layer sostituisce ogni numero negativo del Pooling Layer con il valore zero. Viene spesso utilizzata per il training perché il calcolo della sua derivata è veloce ed il valore che questa assume è zero solo se il valore in input è minore di zero.

Pooling Layer: lo scopo del Pooling Layer è di comprimere (sotto-campionare) l'immagine di input in maniera tale da ridurre il carico computazionale, l'utilizzo di memoria e il numero di parametri della rete; un neurone in un Pooling Layer è connesso agli output di un numero molto limitato di neuroni del livello precedente. Un neurone del Pooling Layer non ha pesi. Per questo livello, per ciascun neurone, è necessario definire: pooling kernel, stride e tipo di padding.

Fully Connected Layer: è solitamente posto alla fine della rete e si occupa di prendere le immagini filtrate ad alto livello e tradurre in categoria ciò che la rete ha analizzato, quindi il suo scopo è quello di utilizzare le feature per classificare le immagini. Si tratta di un Multi-Layer Perceptron che usa una funzione di attivazione Softmax, la quale prende in input un vettore di k numeri reali e li normalizza in una distribuzione di probabilità tirando fuori k valori di probabilità; trasforma i k valori del vettore nel range (0,1) in maniera tale che la sommatoria dei k valori del vettore sia 1.

$$f_i(\vec{a}) = \frac{e^{a_i}}{\sum_k e^{a_k}}$$

Equazione 3.3: funzione di attivazione Softmax

Il termine Fully-Connected indica che ogni neurone del layer precedente è collegato a tutti i neuroni del layer successivo.

In questo progetto suggeriamo di utilizzare una rete neurale convoluzionale per la classificazione dei tipi di battiti ECG sul dataset MIT-BIH.

La *Figura 3.4* mostra l'architettura di rete proposta in tale studio per effettuare la classificazione dei segnali ECG.

[9] https://webthesis.biblio.polito.it/13125/1/tesi.pdf

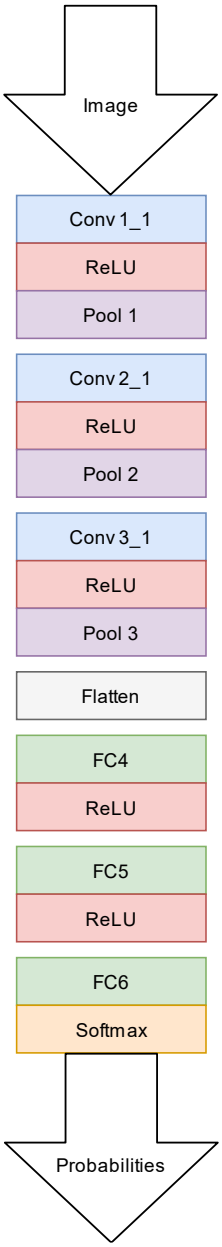

Figura 3.4: architettura CNN

In particolare, l'Input Layer riceve l'immagine caratterizzata da una size di 55x1 (i valori del vettore delle caratteristiche ottenuto a valle del processo di dimensionality reduction), specificata tramite il parametro *shape*.

Successivamente, nel *block 1* abbiamo un Convolutional Layer che utilizza la *Convolution1D* di Keras caratterizzata da 64 *kernel* di *dimensione* 6. Siccome al primo convolutional layer (cioè quello successivo all'input layer) viene fornita l'intera informazione dell'immagine, è stato deciso di applicare dei kernel di dimensione maggiore (6), rispetto a quelli utilizzati nei successivi convolutional layer (3). Questo perché, avendo a disposizione tutta l'informazione, possiamo permetterci di sintetizzare maggiormente l'immagine. Inserendo sempre più convolutional layer, la dimensione del volume dell'immagine diminuirà velocemente; pertanto, non possiamo utilizzare per i kernel dei layer successivi la stessa dimensione, altrimenti si perderebbero troppe informazioni.

Su questo layer viene utilizzata la funzione di attivazione *ReLU* (le motivazioni di tale scelta sono specificate nella descrizione fornita precedentemente).

Successivamente, viene utilizzato un *Batch Normalization Layer* che normalizza e ridimensiona input o attivazioni: quest'ultimo normalizza le attivazioni del livello precedente in ciascun batch, ovvero applica una trasformazione che mantiene l'attivazione media vicino a 0 e la deviazione standard di attivazione vicino ad 1. Si tratta di una tecnica che permette di migliorare velocità, performance e stabilità di una rete neurale. In particolare, questa tecnica permette ad ogni layer di imparare autonomamente ed in maniera indipendente rispetto agli altri layer.

Infine, viene introdotto un *Pooling Layer* e viene utilizzato l'algoritmo *Max-Pooling* come algoritmo di Pooling. Come già accennato precedentemente, l'obiettivo è sotto-campionare una rappresentazione di input (immagine) riducendo la sua dimensionalità e consentendo di fare ipotesi sulle caratteristiche ottenute nelle sotto-regioni. Quindi, solo il massimo valore di input in ciascun kernel va al livello successivo, mentre gli altri input vengono persi. Nel dettaglio, è stato utilizzato un *pooling kernel* di *dimensione* 3 con *stride* 2. La motivazione di tale scelta è che, avendo effettuato un processo di dimensionality reduction, e quindi avendo già perso una parte dell'informazione, utilizzare un pooling kernel di dimensione troppo elevata avrebbe fatto perdere troppa informazione.

Il tipo di *padding* utilizzato è "*Same*". Il padding si riferisce alla quantità di pixel aggiunti ad un'immagine quando viene elaborata dal kernel di una CNN. Nel dettaglio, settando come padding "Same", ogni valore di pixel aggiunto sarà di valore 0. Il padding lavora estendendo l'area di un'immagine che una rete neurale convoluzionale elabora. Per aiutare il kernel ad elaborare l'immagine, viene aggiunto il padding al frame per consentire più spazio al kernel per coprire

l'immagine. L'aggiunta del padding ad un'immagine elaborata da una CNN consente quindi un'analisi più accurata delle immagini.

Il *block 2* e il *block 3* sono uguali al block 1 solo che la Convolution1D è caratterizzata da 64 kernel di dimensione 3.

Successivamente ai 3 blocks viene inserito un *Flatten Layer*. Tale layer converte i dati in un array mono-dimensionale. La ragione per cui lo utilizziamo è perché in seguito avremo bisogno di inserire questi dati in una rete neurale artificiale. Ciò che accade dopo aver effettuato l'operazione di Flattening è di andare quindi ad inserire il vettore dei dati di input nella rete neurale artificiale per elaborarli ulteriormente.

A valle del Flatten Layer vengono inseriti due *Fully Connected Layer*, il primo caratterizzato da 64 *neuroni*, il secondo da 32 neuroni e per entrambi i layer viene utilizzata la funzione di attivazione ReLU. Infine, l'*Output Layer* è formato da 5 *neuroni* e viene utilizzata la funzione di attivazione *Softmax*. La motivazione della scelta del numero di neuroni nell'Output Layer è dettata dal numero di classi da poter associare ad un'istanza data in input. Per quanto riguarda la scelta del numero di neuroni utilizzati negli altri 2 Fully Connected Layer non vi sono particolari motivazioni, esso è stato scelto effettuando il tweaking dei parametri, fino ad ottenere il risultato migliore. Esistono molti metodi empirici per determinare un numero accettabile di neuroni da utilizzare. In particolare, ci siamo focalizzati sul seguente: "il numero di neuroni dovrebbe essere inferiore al doppio della dimensione dell'input layer"[10].

Quando le classi sono esclusive (cioè deve essere fornita in output un'unica classe), l'Output Layer è tipicamente modificato rimpiazzando la funzione di attivazione utilizzata negli strati precedenti (in questo caso ReLU) con una funzione Softmax, in questo caso l'Output Layer viene chiamato *Softmax Output Layer*. Come già descritto precedentemente, tale funzione fornisce in output la probabilità con cui un'istanza in input appartenga ad una delle k classi considerate (in questo caso 5) e in genere, viene associata all'istanza in input la classe per cui è stata calcolata la probabilità più alta.

Per la compilazione del modello, la rete utilizza l'algoritmo di ottimizzazione della *Discesa del Gradiente Adam* utilizzando una funzione di perdita logaritmica chiamata *categorical_crossentropy*. Essa è applicabile quando un'istanza può appartenere ad una sola classe.

La Discesa del Gradiente è un algoritmo di ottimizzazione generico che è capace di trovare una soluzione ottimale per una vasta gamma di problemi. L'idea generale della Discesa del Gradiente è quella di cercare di manipolare in modo iterativo i parametri fino ad arrivare alla minimizzazione della funzione di perdita. L'inclinazione viene calcolata tramite il gradiente, cioè tramite la derivata. Dopo aver calcolato il gradiente, l'algoritmo va a modificare iterativamente i valori dei parametri nella direzione del gradiente discendente in modo da prendere il path che porta ad azzerare il valore della funzione di perdita. Una volta azzerato il gradiente della funzione di perdita si è raggiunto un minimo, dove i punti di minimo sono rappresentati dai punti in cui c'è derivata zero. L'obiettivo del training è quello di avere un modello che tiri fuori un valore elevato di probabilità per la classe di appartenenza di un'istanza e che tiri fuori una bassa probabilità per le altre classi.

La funzione di perdita categorical_crossentropy penalizza il modello quando stima una probabilità bassa per la classe target. Pertanto, tale funzione è usata spesso per misurare quanto le probabilità di appartenenza delle istanze alle varie classi previste dal modello coincidono con le classi target.

Essa è definita come:

$$CCE(p,t) = -\sum_{c=1}^{C} t_{o,c} \log(p_{o,c})$$

Equazione 3.4: funzione di perdita categorical_crossentropy

[10] https://www.heatonresearch.com/2017/06/01/hidden-layers.html

IV. FEATURE EXTRACTION (RGB vs. HSB)

Sono state implementate due tecniche di feature extraction per andare a costruire il vettore delle caratteristiche necessario per descrivere i singoli segnali ECG estratti da fonti video. Quest'ultimo viene poi fornito in input al classificatore in modo tale da assegnare una classe al segnale preso in considerazione. Tuttavia, solo una è stata scelta per l'applicazione a fonti video, cioè quella che permette di ottenere una migliore generalizzazione.

A. Tecnica basata sul modello RGB

Inizialmente, per effettuare il processo di feature extraction dei segnali ECG da fonti video è stata utilizzata una tecnica basata sul modello RGB (Red Green Blue). In particolare, per una data immagine, il vettore delle caratteristiche per un segnale ECG rilevato è stato ricostruito andando a considerare un Marker Point di un determinato colore presente all'interno dell'immagine. Quindi, l'algoritmo comincia ad analizzare l'immagine andando a considerare i pixel del colore di cui è caratterizzato il Marker Point. Rilevato quest'ultimo, l'informazione viene ricavata decrementando la coordinata dell'asse Y fino a trovare il pixel di colore nero che identifica il picco del segnale da estrarre. Fatto ciò, selezionato l'intervallo di valori da considerare, il vettore delle caratteristiche che rappresenta il segnale viene ricostruito, ed è esso che poi viene dato in input al classificatore.

Sono state effettuate delle sperimentazioni utilizzando tale metodo che hanno portato dei buoni risultati. Nel dettaglio, tale metodo è stato applicato a delle immagini recuperate in rete caratterizzate da un Marker Point per ciascun segnale mostrato. Come si può notare, effettuare delle classificazioni in real-time utilizzando tale tecnica risulta essere complesso in quanto si dovrebbe procedere, non solo ad effettuare la rilevazione del segnale, ma anche all'etichettatura di esso, attraverso un Marker Point, per costruire il vettore delle caratteristiche.

Ricapitolando, questa tecnica risulta essere valida ma presenta delle grosse limitazioni poiché ciascun segnale di un video da analizzare dovrebbe essere caratterizzato da un Marker Point. Quindi, questo non permette che tale tecnica possa generalizzare bene su un qualsiasi video.

La *Figura 4.1* mostra un esempio di segnali etichettati con un Marker Point.

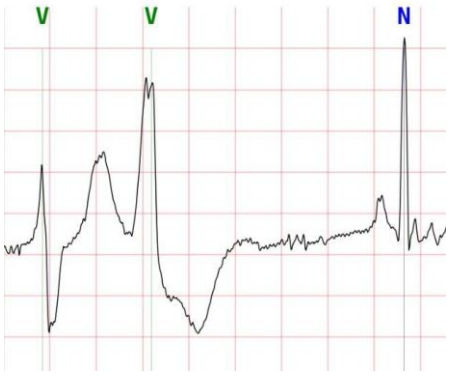

Figura 4.1: esempio di segnali etichettati con un Marker Point

B. Tecnica basata sul modello HSB

Considerando tutti i limiti riportati dal precedente processo di feature extraction illustrato, abbiamo realizzato un nuovo processo. Siamo passati dunque ad un'altra tecnica che generalizzasse la rilevazione del segnale senza Marker Point. In particolare, si è stabilito di non lavorare con il modello RGB per poter effettuare l'estrazione del segnale ECG, ma piuttosto andare a considerare il modello HSB (Hue Saturation Brightness). Questo perché tale modello ci permette di lavorare singolarmente sui tre parametri specificati, funzionalità che non ci viene fornita utilizzando il modello RGB.

Presa un'immagine di ECG:

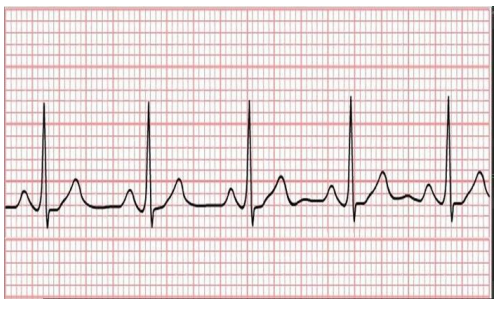

Figura 4.2: immagine ECG

viene effettuata una conversione di modello (RGB → HSB) in modo tale da ottenere il parametro di saturazione su cui ci siamo concentrati. In particolare, andiamo a diminuire la saturazione, e di conseguenza il colore dell'immagine diventa più

tenue e tende al grigio (chiaro). La desaturazione di un'immagine digitalizzata è una delle tecniche con cui si può trasformare un'immagine a colori in una in scala di grigi, ed è proprio quello su cui ci siamo focalizzati. Infatti, trasformando l'immagine a colori in una in scala di grigi, risulta molto semplice estrarre l'ECG poiché il segnale viene messo in risalto rispetto al background. Il risultato che otteniamo applicando tale tecnica all'immagine precedentemente illustrata è il seguente:

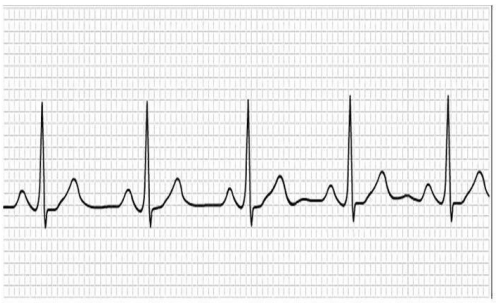

Figura 4.3: immagine saturata

Avendo messo in evidenza il segnale rispetto al background risulta essere molto semplice estrarlo. Il risultato ottenuto su tale immagine è:

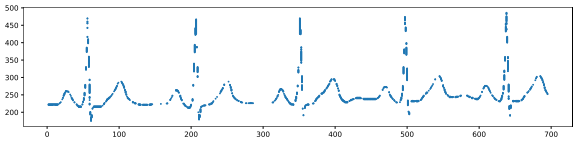

Figura 4.4: segnale estratto

Il passo successivo che si è compiuto è stato quello di andare ad individuare i picchi del segnale ricostruito per poter estrarre i singoli segnali che dovranno poi essere classificati. Estratto il picco di uno dei segnali, a partire da tale posizione viene effettuata una ricostruzione di esso. Ad esempio, il risultato ottenuto ricostruendo il primo segnale utilizzando il picco estratto è:

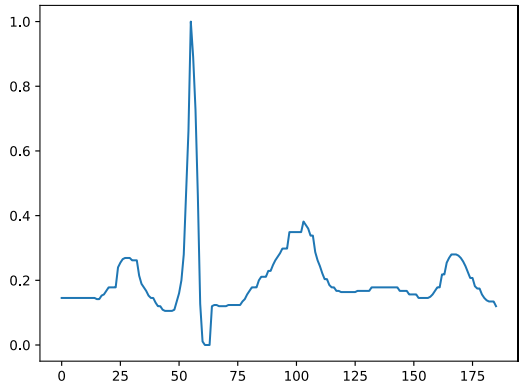

Figura 4.5: singolo segnale ricostruito

Questo è il segnale finale che viene preso in considerazione. Dando in input al classificatore il valore delle feature estratte verrà fornito in output la classe da associare al segnale. Infine, i valori del vettore delle caratteristiche estratto vengono normalizzati nel range (0,1) così come sono caratterizzati i segnali presenti nel dataset MIT-BIH.

Sull'asse delle ordinate vengono rappresentati i valori del vettore delle caratteristiche. Essendo esso mono-dimensionale, per plottare il segnale c'è bisogno di una rappresentazione bi-dimensionale. Pertanto, sull'asse delle ascisse rappresentiamo gli indici del vettore delle caratteristiche (da 0 a 186). Quindi, ad ogni valore dell'asse delle ascisse verrà associato un valore del vettore delle caratteristiche.

V. SPERIMENTAZIONE

A. Strumentazione utilizzata

Per ciò che concerne l'hardware, è stata utilizzata una macchina equipaggiata con processore i5-8250U, accompagnato da una Nvidia Geforce MX150, che permette di velocizzare la risoluzione del problema, rispetto al solo utilizzo della CPU. Le GPU ad alte prestazioni sono dotate, infatti, di un'architettura parallela molto efficiente per il deep learning, al contrario delle CPU. La macchina è stata equipaggiata con sistema operativo Windows 10 e, come ambiente di sviluppo, è stato utilizzato l'IDE Pycharm: un editor molto comodo, veloce ed intelligente che fornisce auto-completamento del codice e suggerimenti in fase di scrittura.

B. *Training del classificatore*

Abbiamo suddiviso il dataset utilizzando l'80% dei dati per il training set e il 20% dei dati per il test set tramite la funzione *train_test_split*. Essa va ad effettuare uno splitting casuale delle istanze del dataset in training set e test set. Inoltre, è stato settato il parametro *random_state* per ottenere la stessa suddivisione su diverse chiamate di funzione.

Quando si modellano problemi di classificazione multi-classe utilizzando reti neurali, è buona norma rimodellare l'attributo dipendente in una matrice con valori booleani per ciascun valore di classe. Questa procedura è chiamata *One-Hot-Encoding*. Per fare ciò è stata utilizzata la funzione *to_categorical*, la quale trasforma il vettore delle classi in una matrice binaria.

Per il training sono state utilizzate 40 *epoche*, che rappresentano il numero di volte in cui il modello è esposto al set di dati di training; mentre il parametro *batch_size* è stato settato a 32, che rappresenta il numero di istanze di allenamento mostrate al modello prima di eseguire l'aggiornamento dei pesi.

Infine, sono state utilizzati due metodi di *callback*, cioè la tecnica dell'*Early Stopping* e il *ModelCheckpoint*.

Un modo diverso per regolarizzare algoritmi iterativi, come ad esempio la Discesa del Gradiente, è quello di arrestare il training non appena l'errore di validazione raggiunge il punto di minimo (cioè il punto a derivata 0). Tale tecnica è appunto chiamata Early Stopping. Dopo un certo numero di epoche l'errore di validazione smette di diminuire e incomincia a crescere, mentre quello del training continua a decrescere, questo significa che il modello dopo tante iterazioni inizia ad adattarsi troppo ai dati di training. Quindi con questa tecnica fermiamo l'addestramento non appena l'errore di validazione raggiunge il punto di minimo.

La tecnica di callback ModelCheckpoint viene utilizzata insieme all'allenamento effettuato tramite il metodo *fit* per salvare un modello o pesi ad un certo istante di tempo. In questo caso è stata utilizzata l'opzione di andare a salvare il modello "migliore", dove per migliore intendiamo il modello che minimizza la funzione di perdita. Settando il parametro *save_best_only* a True l'ultimo modello "migliore" in base alla funzione di perdita monitorata non verrà sovrascritto.

C. *Adattamento a fonti video ed in real-time*

Il processo di Feature Extraction precedentemente illustrato è stato utilizzato per poter andare ad effettuare la classificazione di segnali ECG estratti da fonti video ed in real-time. Tali operazioni sono state implementate tramite l'utilizzo della libreria OpenCV. Essa è una libreria software open-source che opera nell'ambito della visione artificiale, permettendo l'analisi di video sia da fonti statiche che da fonti in tempo reale.

Preso in input il path di un video, si procede ad operare sui singoli frame andando ogni volta ad estrarre il primo segnale presente nel frame, utilizzando la tecnica basata sul modello HSB; il vettore delle caratteristiche costruito viene fornito in input al classificatore che stima una probabilità con cui il segnale appartenga a ciascuna delle classi. Tramite la funzione *argmax* andiamo a determinare qual è la classe per la quale è stata stimata la probabilità più alta, che è la classe da associare al segnale. Tale etichetta viene rappresentata sul segnale, insieme alla probabilità stimata per essa. L'operazione viene effettuata per ogni singolo frame estratto dal video e quindi come output finale viene fornito il video etichettato.

Per quanto riguarda la classificazione in real-time, mostrando al dispositivo di acquisizione video utilizzato il segnale ECG da etichettare, viene effettuato lo stesso procedimento illustrato precedentemente però mostrando l'etichetta direttamente sullo stream video.

La *Figura 5.1* mostra un esempio di etichettatura di un segnale.

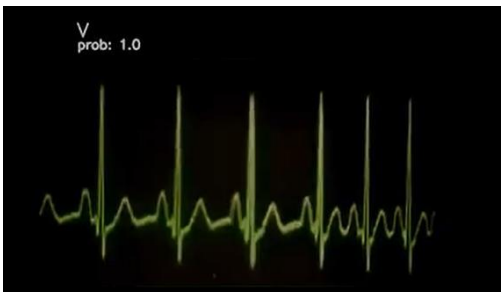

Figura 5.1: esempio di etichettatura di un segnale

VI. RISULTATI

In tale sezione vengono illustrati i risultati ottenuti nelle varie fasi del progetto, in particolare quelli raggiunti applicando i processi di resampling e feature selection al dataset e quelli relativi alle perfomance del modello.

A. Risultati resampling

Per analizzare le performance di generalizzazione del modello sono state utilizzate le *Learning Curves*. Una Learning Curve è un diagramma che mostra le performance di apprendimento di un modello che progressivamente avanza col training sul dataset. Esse sono uno strumento diagnostico ampiamente utilizzato nel Machine Learning per algoritmi che apprendono in modo incrementale da un set di dati di training.

Il modello può essere valutato sul training set e sul test set dopo ogni epoca durante il training e possono essere creati grafici delle performance misurate per mostrare le Learning Curves. Generalmente dovremmo ottenere che le performance del modello migliorano all'aumentare del numero di istanze di training prese in considerazione. La revisione di tali strumenti diagnostici dei modelli durante l'allenamento può essere utilizzata per diagnosticare problemi con l'apprendimento, ad esempio verificare se un modello va in *overfitting*.

L'overfitting si riferisce a un modello che ha appreso troppo bene l'insieme di dati di training, incluso il rumore statistico o le fluttuazioni casuali. Il problema dell'overfitting è che più il modello si adatta ai dati di training, meno è in grado di generalizzare su nuovi dati, con conseguente aumento dell'errore di generalizzazione. Un diagramma delle Learning Curves mostra un overfitting se la curva della validation loss diminuisce fino ad un certo punto e ricomincia ad aumentare[11].

La *Figura 6.1* delle Learning Curves ottenute dal training del modello sul dataset sbilanciato mostra un evidente overfitting, poichè esiste un divario consistente tra l'errore che il modello commette sulle istanze del training set e quelle del test set.

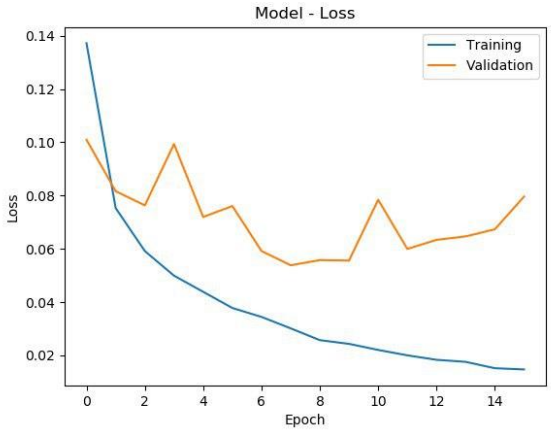

Figura 6.1: Learning Curves ottenute effettuando il training del modello sul dataset sbilanciato

Quindi, man mano che il modello effettua il training su un numero di istanze più elevato, riesce ad approssimare sempre meglio i dati di training; questo è il motivo per cui la curva che rappresenta l'errore commesso dal modello sulle istanze del training set decrementa sempre di più. Allo stesso tempo, quindi, riesce ad estrarre una conoscenza sempre più accurata, ottenendo delle performance di generalizzazione migliori, e quindi anche la validation loss comincia a decrementare.

Se ad un certo punto la curva che rappresenta l'errore di generalizzazione del modello cresce nuovamente, mentre quella che rappresenta l'errore commesso dal modello sul training set continua a decrescere, questo indica che il modello peggiora le performance di generalizzazione, poiché effettua overfitting dei dati di training. Questo è esplicitato dal fatto che la curva della validation loss, a partire dalla 11-esima epoca, ricomincia a crescere.

Pertanto, ciò vuol dire che il modello si è adattato troppo ai dati di training e non riesce a generalizzare bene su nuove istanze che non ha mai visto durante la fase di training.

Dall'analisi effettuata possiamo procedere dunque, come accennato nella sezione **III.B**, ad effettuare il resampling del dataset.

La *Figura 6.2* mostra le performance del modello avendo effettuato il training sul dataset bilanciato.

[11] https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance

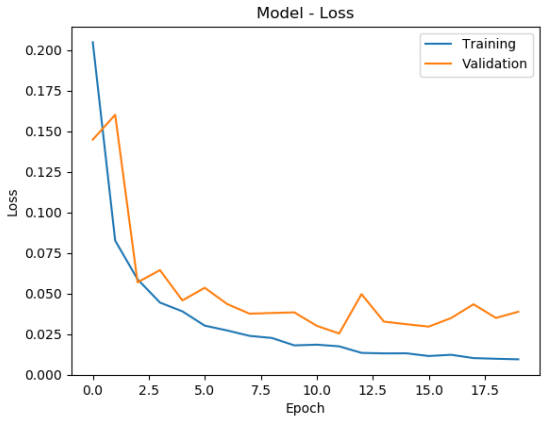

Figura 6.2: Learning Curves ottenute effettuando il training del modello sul dataset bilanciato

Si può notare come non ci sia presenza di overfitting poiché il modello tende a comportarsi allo stesso modo sia sulle istanze di training che sulle istanze usate per il testing, in quanto l'errore commesso risulta essere pressoché simile (tale informazione viene sottolineata dal fatto che l'area che separa le due curve è piccola) e inoltre il plateau (punto in cui il modello non cambia il suo comportamento sulle istanze del training set e del test set anche se esegue nuove epoche) si è raggiunto ad un livello di errore molto basso. Pertanto, otteniamo una Learning Curve ideale poiché le curve di apprendimento del test e del training convergono a valori simili; il divario è minore e quindi la generalizzazione del modello risulta essere migliore.

B. Risultati Feature Selection

Per poter effettuare un confronto tra i processi di training effettuati sul training set considerando e non le feature meno rilevanti individuate tramite le FD, sono stati presi in considerazione i seguenti parametri:

- Accuracy, cioè il numero totale di istanze classificate correttamente diviso per il numero totale di istanze classificate;
- Tempo impiegato per effettuare il training.

Come accennato nella sezione **III.C** per la rimozione delle feature è stato utilizzato un approccio bottom-up.

Avendo già ottenuto buoni risultati rimuovendo tutte le feature individuate tramite le FD, non si è ritenuto necessario andare ad analizzare altre combinazioni di colonne.

Effettuando il training del modello considerando tutte le feature del dataset, i risultati ottenuti sono:

- Accuracy: **99,41%**;
- Tempo impiegato per il training: **2220s**.

Effettuando il training del modello rimuovendo dal dataset tutte le feature meno rilevanti, individuate utilizzando le FD estratte, i risultati ottenuti sono:

- Accuracy: **98,47%**;
- Tempo impiegato per il training: **1020s**.

Siccome rimuovendo tutte le feature meno rilevanti il tempo necessario per effettuare il training si riduce drasticamente e, allo stesso tempo, si ottiene un'accuracy di poco minore a quella ottenuta effettuando il training utilizzando tutte le feature (più piccola circa dell'1%), si è stabilito di rimuovere dal dataset, e quindi di non considerare, tali feature. Questo perché si è raggiunto il giusto compromesso tra tempo necessario per il training e perdita di informazioni.

Quindi è stato preso in considerazione il dataset composto da 56 feature.

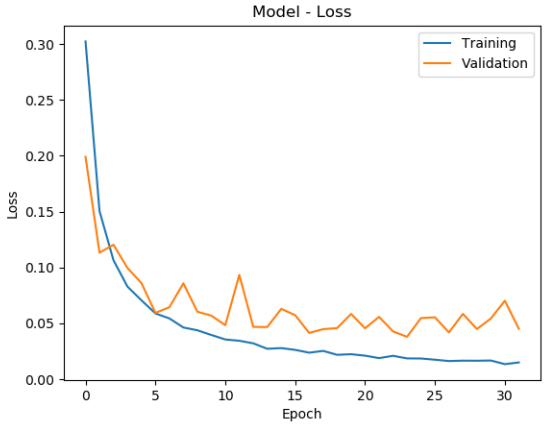

Figura 6.3: Learning Curves ottenute effettuando il training del modello sul dataset bilanciato e avendo rimosso le feature meno rilevanti

Osservando la *Figura 6.3* delle Learning Curves, ottenute effettuando il training del modello sul dataset risultante dal resampling e, avendo rimosso tutte le feature meno rilevanti individuate tramite l'utilizzo delle FD, considerando le stesse analisi fatte in precedenza, possiamo prendere in considerazione il modello risultante poiché non presenta overfitting.

C. Confusion Matrix

Nel campo dell'apprendimento automatico e in particolare del problema della classificazione statistica, una confusion matrix è un layout di tabella specifico che consente di ricavare una serie di parametri utili per descrivere le performance di un classificatore. In particolare, l'output di una confusion matrix per una classificazione binaria è il seguente:

	1 (Predicted)	0 (Predicted)
1 (Actual)	True Positive	False Negative
0 (Actual)	False Positive	True Negative

Possiamo quindi osservare come le true classes (classi supervisionate) sono poste sulle righe della matrice, mentre le predicted classes sono poste sulle colonne. Nelle celle della matrice vengono specificati:

- True Positive, numero di istanze positive classificate come positive dal modello;
- False Positive, numero di istanze negative classificate come positive dal modello;
- False Negative, numero di istanze positive classificate come negative dal modello;
- True Negative, numero di istanze negative classificate come negative dal modello;

Nel nostro caso, l'output della confusion matrix è una matrice di cinque righe e cinque colonne, poiché le classi da prevedere sono cinque. Ogni cella mostra la percentuale con cui un campione associato ad una determinata predicted class è stato classificato come la true class corrispondente. La confusion matrix è stata utilizzata per valutare le performance della rete neurale convoluzionale.

	N	S	V	F	U
N	0.97	0.02	0.008	0.0052	0.00075
S	0.0027	1	0.0005	0.0005	0
V	0.0017	0.00025	0.99	0.004	0.001
F	0.001	0	0	1	0
U	0.0025	0.00075	0.0005	0	1

Tabella 6.1: Confusion Matrix

Come si può osservare dalla *Tabella 6.1* gli errori commessi sono bassi, in particolare la classe N è quella per la quale viene commesso l'errore percentuale di previsione più elevato.

A partire dalla confusion matrix, attraverso l'utilizzo della funzione *classification_report* della libreria sklearn, è possibile creare un text report che mostri le principali metriche di classificazione, illustrate nella *Tabella 6.2* e nella *Tabella 6.3*.

	Precision	Recall	F1-score	Support
0	1.00	0.97	0.98	4018
1	0.98	1.00	0.99	4015
2	0.99	0.99	0.99	4002
3	0.99	1.00	1.00	3987
4	1.00	1.00	1.00	3978

Tabella 6.2: text report

	Precision	Recall	F1-score	Support
micro avg	0.99	0.99	0.99	20000
macro avg	0.99	0.99	0.99	20000
weighted avg	0.99	0.99	0.99	20000
samples avg	0.99	0.99	0.99	20000

Tabella 6.3: text report

La *precision* di una classe definisce quanto sia affidabile il risultato quando il modello associa quella classe ad un'istanza, mentre la *recall* di una classe esprime quanto bene il modello sia in grado di rilevare quella classe. Entrambe le metriche assumono un valore compreso nel range (0,1). Il *support* indica, nella Tabella 6.2, il numero di istanze per ciascuna classe che ricade nel test set, mentre nella Tabella 6.3 indica il numero totale di istanze presenti all'interno del test set.

Per una data classe, le diverse combinazioni di recall e precision hanno i seguenti significati:

- recall elevata + precision elevata: la classe è perfettamente gestita dal modello;
- recall bassa + precision elevata: il modello non è in grado di rilevare bene la classe ma è altamente affidabile quando lo fa;
- recall elevata + precision bassa: la classe è ben rilevata ma il modello include anche istanze di altre classi al suo interno;
- recall bassa + precision bassa: la classe è gestita male dal modello.

Come si può notare dalla *Tabella 6.2*, sono stati ottenuti valori di precision e recall tendenti ad 1 per ciascuna classe; questo significa che il modello permette di minimizzare contemporaneamente sia il numero di falsi positivi (avendo ottenuto valori di precision vicini ad 1) che il numero di falsi negativi (avendo ottenuto valori di recall vicini ad 1) per ognuna di essa. Allo stesso tempo, quindi, otteniamo valori elevati anche della metrica *f1-score*, poiché essa risulta essere la media armonica della precision e della recall. Pertanto, il modello addestrato è caratterizzato da una buona capacità di generalizzazione su tutte le classi.

D. ROC Curve

Oltre alla confusion matrix, è stata presa in considerazione anche la ROC curve; esso è uno schema di rappresentazione grafica che permette di valutare la bontà di un classificatore multi-classe. Infatti, in molte applicazioni sperimentali si presenta l'esigenza di operare con classificatori multi-classe che possano attribuire, dall'analisi dei dati, la classe di appartenenza ad una data istanza analizzata. Definite le cinque classi che possono essere associate ad un segnale, definiamo "true positive" il numero di casi in cui il classificatore identifica correttamente la classe di un'istanza, e "false positive" il numero di casi in cui il classificatore sbaglia la predizione.

La ROC curve viene costruita considerando tutti i possibili valori del test e, per ognuno di questi, si calcola la proporzione di true positive e la proporzione di false positive. Congiungendo i punti che mettono in rapporto la proporzione di true positive e di false positive (le cosiddette coordinate) si ottiene una curva chiamata ROC curve per ogni classe. L'area sottostante alla ROC curve (AUC, acronimo dei termini inglesi "Area Under the Curve") è una misura di accuratezza diagnostica. Si considera adeguato un test diagnostico con un'area sotto la curva $\geq$ 80%. Tanto maggiore è l'area sotto la curva (cioè tanto più la curva si avvicina al vertice in alto a sinistra del grafico) tanto maggiore è il potere discriminante del test.

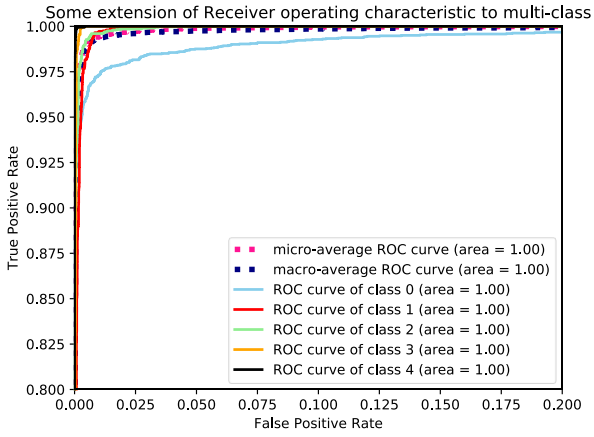

Figura 6.4: ROC Curve

Solitamente, quando si vuole ottenere un classificatore che abbia un valore di recall elevato per ogni classe, quindi che vada a minimizzare il numero di falsi negativi, bisogna accettare il fatto

che venga introdotto un certo numero di falsi positivi, in quanto, all'aumentare dei valori della recall, otteniamo valori di precision più bassi. La ROC Curve viene utilizzata proprio per analizzare il compromesso tra il true positive rate (anche detto recall) e il false positive rate; un classificatore che si comporta correttamente dovrebbe ottenere valori di recall elevati, minimizzando allo stesso tempo il numero di falsi positivi.

Come si evince dalla *Figura 6.4*, la ROC Curve aumenta rapidamente da 0 a 1 per ciascuna classe; ciò indica che per ognuna di essa riusciamo ad ottenere valori di recall elevati, ottenendo allo stesso tempo pochi falsi positivi; pertanto, ciò esplicita il fatto che il classificatore riesce a generalizzare bene su ogni classe.

VII. CONCLUSIONI

In questo studio abbiamo presentato un metodo per la classificazione del battito cardiaco ECG basandoci sulle istanze presenti nel dataset MIT-BIH. Tramite l'utilizzo delle FD siamo stati capaci di ridurre la dimensionalità del dataset, andando a determinare quali sono le feature più rilevanti per le previsioni, potendo conservare la maggior parte delle informazioni e andando a decrementare di molto il tempo necessario per effettuare il training del modello sul dataset. In particolare, abbiamo addestrato una rete neurale convoluzionale che ha fornito un'accuracy del 98.47%. Inoltre, è stato proposto un nuovo metodo di feature extraction basato sul modello HSB per poter estrarre il vettore delle caratteristiche da associare ad un segnale considerato. Tale tecnica è stata utilizzata per poter andare ad effettuare la classificazione di segnali ECG da fonti video ed in real-time. Essa però pone delle limitazioni sul fatto di dover utilizzare una telecamera ad alte prestazioni per una migliore interpretazione del segnale.

VIII. SVILUPPI FUTURI

Uno degli sviluppi futuri su cui ci si potrà focalizzare, è quello di andare ad incrementare ulteriormente il numero di istanze all'interno del dataset, in modo tale da poter migliorare l'accuracy e quindi la capacità di generalizzazione del modello. Inoltre, per ridurre maggiormente il numero di feature del dataset da utilizzare, si potrebbe pensare anche di andare a ricavare delle *Dipendenze Funzionali Rilassate* (RFD) sul dataset, in modo tale da andare a ridurre ulteriormente il tempo necessario per effettuare il training del modello. Allo stesso tempo, le RFD potrebbero essere utilizzate per effettuare un confronto con i risultati ottenuti tramite l'utilizzo delle FD. Infine, un altro punto da prendere in considerazione potrebbe essere quello di andare a raffinare il processo di feature extraction in modo tale che esso possa essere applicato anche in situazioni in cui viene utilizzata una telecamera con bassa risoluzione, limite che si pone attualmente.

REFERENZE

[1] Goldberger Ary, Amaral Luis N,, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23): e215-e220

[2] Mohammad Kachuee, Shayan Fazeli, Majid Sarrafzadeh, University of California, Los Angeles (USA): ECG Heartbeat Classification: A Deep Transferable Representation

[3] Qibin Zhao, Liqing Zhang, Department of Computer Science and Engineering Shanghai Jiaotong University, Shanghai, 200030, China: ECG Feature Extraction and Classification Using Wavelet Transform and Support Vector Machines

[4] Haotian Shi, Haoren Wang, Yixia Huang, Liqun Zhao, Chengjin Qin, Chengliang Liu: A hierarchical method based on weighted extreme gradient boosting in ECG heartbeat classification

[5] Fajr Ibrah Alars and Younes J: Analysis and classification of heart diseases using heartbeat features and machine learning algorithms

[6] Yang Zhou et al 2020 IOP Conf.: Earth Environ. Sci. 428 012014: ECG Heartbeat Classification Based on ResNet and Bi-LSTM

[7] Moody and R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database," in *IEEE* Engineering in Medicine and Biology Magazine, vol. 20, no. 3, pp. 45-50, May-June 2001, doi: 10.1109/51.932724