UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA BIOMEDICA

Classificazione di segnali ECG tramite tecniche di deep learning

Relatore
Prof. Castellaro Marco

Correlatore
Dott. Scarpa Fabio

Laureando
Gorghetto Rachele

ANNO ACCADEMICO 2024-2025

Data di laurea 17/03/2025

Al mio portatile,
perché non si è mai arreso.

DAI! DAI! DAI! - René Ferretti

Abstract

In questa tesi si espone il progetto svolto durante il tirocinio nell'azienda M31, il cui obiettivo era di costruire un modello di deep learning (di tipo LSTM) in grado di distinguere dei segnali ECG su 12 derivazioni, di un database privato in possesso di dell'azienda, in sani e patologici.

La tesi è composta da una introduzione sul segnale ECG e su come viene classificato in letteratura scientifica usando l'intelligenza artificiale, da una presentazione dei database utilizzati, dai modelli costruiti e dalla discussione dei risultati ottenuti, oltre che da una analisi delle difficoltà e limitazioni riscontrate e da spunti sul futuro di questi approcci.

Dall'applicazione dei modelli addestrati su un database pubblico, si è riscontrata una accuratezza di oltre l'85% sui segnali ECG di M31, ma questa va verificata utilizzando dei dati con delle diagnosi verificate e con un addestramento esclusivo su un database privato molto più vasto.

Indice

1 Introduzione **1**
1.1 Il segnale ECG 1
1.2 Classificazione di segnali ECG tramite IA 3

2 I database **5**
2.1 Il database privato 5
2.2 Il database pubblico 6
2.3 Suddivisione delle patologie nelle classi di predizione 6

3 Sviluppo dell'algoritmo **7**
3.1 Implementazione 7
3.1.1 Librerie utilizzate 7
3.1.2 Caricamento e pre-elaborazione dei dati 7
3.1.3 Modello IA 10
3.2 Metriche utilizzate 13
3.3 Modalità di testing 13
3.3.1 Cross validation 14
3.3.2 Addestramento sul dataset pubblico 15

4 Risultati **17**
4.1 Database privato 17
4.1.1 Modello senza il layer di BatchNormalization 17
4.1.2 Modello con il layer di BatchNormalization 17
4.2 Database pubblico 19
4.2.1 Modello senza il layer di BatchNormalization 19
4.2.2 Modello con il layer di BatchNormalization 19

5 Discussione dei risultati **23**
5.1 Database privato 23

5.2 Database pubblico 25

6 Conclusione **27**

6.1 Difficoltà riscontrate 27

6.2 Aspetti futuri 28

Elenco delle figure

1.1 Confronto tra due ECG 2
1.2 Onda di un ciclo cardiaco normale (immagine presa da [5]) 2

3.1 Modello di una rete LSTM bidirezionale che mostra i layer di ingresso e uscita. Le frecce rosse indicano percorso della sequenza a ritroso e quella verde in avanti.[19] 11
3.2 Struttura del modello 12
3.3 Struttura del modello con il layer di BatchNormalization 12

4.1 Matrici di confusione 19

5.1 Accuratezza 96° ciclo del primo modello 4.1 - dataset privato 23
5.2 Loss 96° ciclo del primo modello 4.1 - dataset privato 24
5.3 Accuratezza 83° ciclo del secondo modello 4.2 - dataset privato 24
5.4 Loss 83° ciclo del secondo modello 4.2 - dataset privato 24
5.5 Accuratezza sul dataset pubblico 4.3 25
5.6 Loss sul dataset pubblico 4.3 25
5.7 Accuratezza sul dataset pubblico con il layer di BatchNormalization 4.4 26
5.8 Loss sul dataset pubblico con il layer di BatchNormalization 4.4 26

Elenco delle tabelle

2.1 Classi database privato 5
2.2 Traduzione della tabella presente in [11] 6
2.3 Distribuzione dati utilizzati 6

4.1 Risultati 96° ciclo del primo modello - dataset privato 18
4.2 Risultati 83° ciclo del secondo modello - dataset privato 18
4.3 Risultati del primo modello - dataset pubblico 20
4.4 Risultati del secondo modello - dataset pubblico 21

Capitolo 1

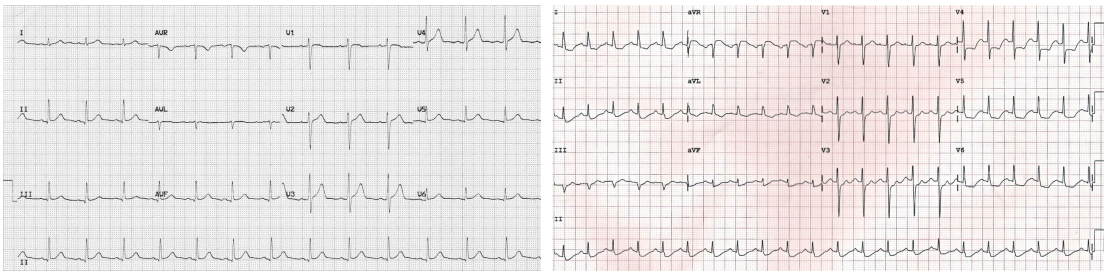
Introduzione

Al giorno d'oggi, da ogni paziente si ricavano moltissimi dati che il medico deve analizzare per poter dare una diagnosi.
Con la diffusione della telemedicina e con i, già disponibili, dispositivi indossabili, in futuro questi dati saranno ancora più abbondanti, quindi, per evitare che il medico venga sovraccaricato, è importante costruire degli strumenti che gli siano di supporto analizzando a priori questi segnali.

É così che, tramite il tirocinio presso l'azienda M31 [1], si è voluto costruire un modello di intelligenza artificiale (IA) per la classificazione di segnali elettrocardiografici (ECG), che è l'oggetto di questa tesi.
Per poter affrontare questo progetto e costruire l'algoritmo, è quindi importante comprendere che cos'è il segnale ECG (1.1) e come viene elaborato e classificato tramite IA in letteratura scientifica (1.2).

1.1 Il segnale ECG

L'elettrocardiogramma è un esame diagnostico che consiste nella registrazione dell'attività elettrica cardiaca, sulla superficie del corpo, tramite elettrodi.
Il segnale ECG standard è registrato tramite 12 elettrodi, tre frontali (I, II, III), tre unipolari aumentati (aVR, aVL, aVF) e sei precordiali per l'esplorazione in profondità (V1, V2, V3, V4, V5 e V6).
Il tracciato che ne deriva è caratterizzato da picchi, onde, complessi e ritmi che, se anormali, determinano specifiche patologie cardiache.



(a) ECG sano [2] (b) ECG infarto NSTEMI inferolaterale [3]

Figura 1.1: Confronto tra due ECG

Tra questi si individuano[1]:

- l'onda P che è prodotta dalla depolarizzazione atriale;
- il complesso QRS prodotto dalla depolarizzazione ventricolare;
- l'onda T prodotta dalla ripolarizzazione ventricolare;
- l'intervallo P-R causato dal ritardo della conduzione nel nodo atrioventricolare;
- il segmento S-T che è legato alla durata media del plateau di alcune cellule ventricolari;
- l'intervallo R-R che individua un ciclo cardiaco e, per questo motivo, viene utilizzato per il calcolo della frequenza cardiaca.

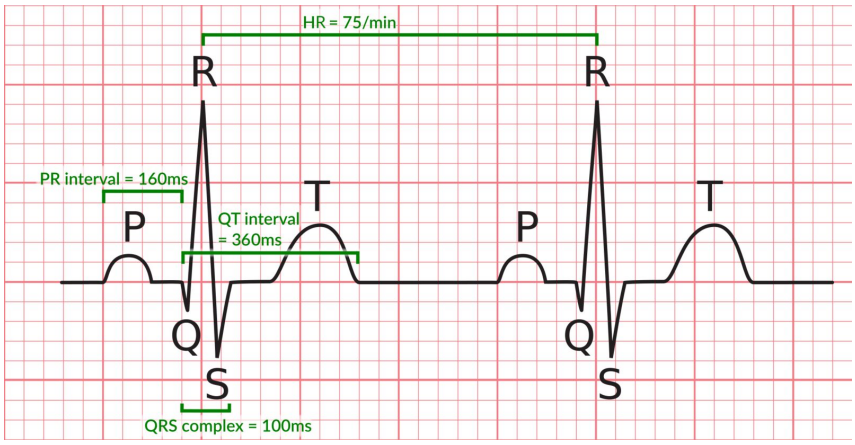


Figura 1.2: Onda di un ciclo cardiaco normale (immagine presa da [5])

[1] tratto da *Medical Instrumentation: Application and Design* [4]

1.2 Classificazione di segnali ECG tramite IA

L'analisi automatica dei segnali ECG non è una richiesta recente, ma è solo negli ultimi anni che ci si sta rivolgendo al machine learning [6].
A partire da metà del '900, si sono costruiti algoritmi che estraggono caratteristiche rilevanti per supportare l'analisi medica, come il calcolo del battito cardiaco misurando gli intervalli RR e le misure specifiche dei singoli picchi, intervalli e complessi. Questi programmi, però, si sono rivelati limitati data la soggettività dei parametri che indicano un buono stato di salute.
Con l'avvento delle nuove tecnologie e la crescita dei segnali da analizzare (grazie anche alla diffusione dei dispositivi indossabili), ci si è rivolti al machine learning, in particolare al deep learning. Questo, infatti, analizzando una grande mole di dati, è in grado di riconoscere caratteristiche comuni autonomamente e utilizzarle per la classificazione.

Per quanto riguarda l'applicazione ai segnali ECG, nel caso della classificazione di aritmie cardiache, tra i diversi metodi di deep learning disponibili, le reti neurali convoluzionali (CNN) sono le più adatte per l'estrazione di feature mentre quelle ricorrenti per la classificazione, come descritto nell'articolo *A review on deep learning methods for ECG arrhythmia classification* [7].
Tra le reti ricorrenti, la Long Short-Term Memory (LSTM) presenta i seguenti vantaggi:

- riesce ad analizzare il segnale ECG senza che questo sia preprocessato [8];
- è in grado di elaborare segnali lunghi senza cadere nel problema della scomparsa del gradiente [9];
- hanno un basso costo computazionale rendendoli adatti alle applicazioni in tempo reale [10].

Capitolo 2

I database

Per poter costruire un modello di classificazione binaria specifica per i dati di M31, sono necessari tanti segnali acquisiti dallo stesso dispositivo o, come in questo caso, altrettanti segnali che abbiano le stesse caratteristiche fondamentali come il numero di derivazioni del segnale ECG.
Per la discussione seguente si indicherà con ”database privato” l’insieme dei dati in possesso di M31 e con ”database pubblico” il dataset PTB-XL ECG [11].

2.1 Il database privato

É composto da 114 segnali ECG su 12 derivazioni, campionati con una frequenza di 1000 Hz, misurati in µV e della durata di 10 secondi.
I segnali, ottenuti con il dispositivo WAM PC di Mortara [12], sono salvati in formato DICOM e, oltre a contenere i dati dell’ECG, presentano anche informazioni anonime sulla diagnosi, l’età e il sesso del paziente.
La diagnosi registrata è stata scritta automaticamente dal dispositivo e non è stata verificata da alcun medico. Le diagnosi sono composte sia da una diagnosi specifica che dalle descrizioni elencate in tabella 2.1.

Numero dati	Classificazione
27	ECG NORMALE
25	ECG AI LIMITI DELLA NORMA
40	ECG ANORMALE
17	RITMO ECG ANORMALE
5	TEST CRITICO

Tabella 2.1: Classi database privato

2.2 Il database pubblico

É composto da 21799 segnali ECG su 12 derivazioni, campionati con una frequenza di 500 Hz (ma resi disponibili anche sottocampionati a 100 Hz), misurati in mV e della durata di 10 secondi. I segnali sono salvati in formato WaveForm DataBase (WFDB) a cui sono legate informazioni anonime riguardo il paziente (sesso, età, altezza, peso) e la diagnosi, che è stata verificata da almeno un cardiologo.
Le diagnosi sono sia specifiche che descritte dalle 5 super classi (non esclusive) presenti in tabella 2.2.

Numero dati	Super classe	Descrizione
9514	NORM	ECG normale
5469	MI	Infarto miocardico acuto
5235	STTC	Variazioni onda ST/T
4898	CD	Disturbi della conduzione
2649	HYP	Ipertrofia

Tabella 2.2: Traduzione della tabella presente in [11]

2.3 Suddivisione delle patologie nelle classi di predizione

Per poter uniformare le classi di diagnosi dei due database, queste sono state divise in due macro classificazioni: Sani (0) e Patologici (1).
Nel database privato sono stati classificati come Sani le diagnosi ”ECG NORMALE” e ”ECG AI LIMITI DELLA NORMA”, come Patologici quelli con diagnosi ”ECG ANORMALE” e ”RITMO ECG ANORMALE” e sono stati esclusi i cinque ”TEST CRITICO” in quanto assenti di una diagnosi chiara.
Nel database pubblico i segnali, ridotti a 19601 in quanto si sono utilizzati solo quelli dedicati al training, sono classificati come Sani quando le diagnosi contengono ”NORM”, mentre come Patologici tutte le altre combinazioni.

Database	Sani	Patologici	Totale
Privato	52	57	109
Pubblico	8551	11050	19601

Tabella 2.3: Distribuzione dati utilizzati

Capitolo 3

Sviluppo dell'algoritmo

3.1 Implementazione

Per sviluppare e implementare l'algoritmo di classificazione binaria si è utilizzato il linguaggio di programmazione Python e si sono prodotti due codici: uno utilizzando esclusivamente i dati privati e l'altro utilizzando il dataset pubblico per il training e quello privato per il testing.

3.1.1 Librerie utilizzate

Per questo progetto si sono utilizzate diverse librerie tra cui:

- *TensorFlow*, per costruire la rete neurale tramite i vari layer e per poter valutare le metriche del modello;
- *Scikit-learn*, per poter utilizzare e visualizzare le matrici di confusione;
- *ECG_R*, ovvero una libreria costruita appositamente per questo progetto per poter accedere a e classificare i segnali ECG privati.

3.1.2 Caricamento e pre-elaborazione dei dati

Dataset privato

Per caricare e suddividere nelle due classi i segnali, si è creata una libreria apposita: ECG_R. Per questa si sono usati Pydicom per la lettura del formato DICOM e Numpy per poter utilizzare e trattare gli array.

```
from pydicom import dcmread
import numpy as np
```

Questa libreria contiene la classe ECG che è composta da due metodi:

- *ecg_label* che assegna la corretta macrodiagnosi;
- *ecg* che restituisce la lista contenete il segnale ECG.

Quando la classe viene evocata per la prima volta questa legge il dato in ingresso, che deve essere in formato DICOM, ed estrae, nella variabile *arr*, il solo segnale ECG convertito in mV.

```
class ECG():
    def __init__(self,source):
        self.ds = dcmread(source)
        self.arr = (self.ds.waveform_array(0))/1000
```

Il metodo ecg_label serve ad assegnare la classificazione Sano, indicato dalla cifra 0, e Patologico, indicato con 1.
Per fare ciò bisogna accedere alle informazioni della diagnosi utilizzando i codici esadecimali identificativi del file DICOM [13] che, nel caso degli ECG a 12 derivazioni, sono (0040,B020) (Waveform Annotation Sequence Attribute) per accedere alla sequenza di annotazioni e (0070,0006) (Unformatted Text Value Attribute) per accedere alla diagnosi testuale.
L'elemento k, che viene incrementato durante il ciclo, serve per scorrere le varie righe di diagnosi fino ad individuare la macro diagnosi o, nel caso non vi sia, l'ultima riga di descrizione e classificarli come assenti di diagnosi (indicati con 999).

```
    def ecg_label(self):
        k=1
        while k<6:
            dato=(self.ds['0040','B020'][k]['0070','0006'].value)
            if dato=='ECG NORMALE' or dato=='ECG AI LIMITI DELLA NORMA':
                return(0)
            elif dato=='ECG ANORMALE' or dato=='RITMO ECG ANORMALE':
                return (1)
            elif dato=='REPORT NON CONFERM.':
                return(999)
            else:
                k+=1
        return(999)
```

Il metodo *ecg* serve per restituire il segnale ECG numerico come una lista.

```
    def ecg(self):
        return(self.arr.tolist())
```

Inoltre, per poter confrontare i segnali privati con quelli pubblici, i primi sono stati sottocampionati a 100 Hz mediando 5 campioni alla volta.

```
    s=[]
    r=0
    for i in range(1000):
        s.append([])
        for l in range(12):
            media_campioni=0
            if r+5<10000:
                for t in range(5):
                    media_campioni+=segn[r+t][l]
                media_campioni=(media_campioni)/5
                s[i].append(media_campioni)
        r+=10
```

Dataset pubblico

Per importare il database pubblico si è utilizzato il codice ”example_physionet”, indicato nella pubblicazione[11], scegliendo di importare i segnali ECG campionati con frequenza di 100Hz dedicati al training.
Per quanto riguarda la divisione nelle due classi, si è assegnato Sano (0) alle macro diagnosi con descrizione contenente ”NORM”, mentre Patologico (1) alle altre combinazioni.
La parte di codice che è stata aggiunta per la classificazione risulta:

```
g_id_train=list(Sy_train.index)
train=[]
r i in (ecg_id_train):
  lab=Sy_train[i]
  if 'NORM' in lab:
      Y_train.append(0)
  else:
      Y_train.append(1)
```

3.1.3 Modello IA

Il modello utilizzato è stato costruito partendo dal codice proposto da Matlab per la classificazione di segnali ECG tramite una rete neurale di tipo LSTM [14]:

```
model=Sequential()
model.add(Input(shape=(1000,12)))
model.add(Bidirectional(LSTM(36)))
model.add(Dense(36))
model.add(Dense(1,activation='sigmoid'))
```

Il tipo di modello e gli strati di questo si sono scelti per le loro caratteristiche descritte qui in seguito:

Sequential [15]

Serve a creare un modello composto da una stratificazione sequenziale di layer.

Input [16]

Impone le dimensioni di ingresso tramite una matrice. In questo caso essa ha le seguenti dimensioni:

- 1000 righe nel caso di segnali a 100Hz (10000 nel caso dei segnali ECG privati originali a 1000Hz), in cui si trovano i dati del segnale ECG;
- 12 colonne che rappresentano le 12 derivazioni.

LSTM [17]

Si è scelta di utilizzare una rete neurale di tipo Long Short-Term Memory in quanto essa è una rete ricorrente che permette di fare confronti con i dati passati e quelli presenti, rendendola quindi adatta alle applicazioni temporali. In questo caso, non ricercando una patologia specifica, ma una distinzione più generale, si è scelto di utilizzarla per poter tenere in considerazione tutto l'andamento del segnale e non solo alcune caratteristiche.
Il numero 36[1] indica il numero di unità in uscita dal layer.

[1] Il numero di unità è stato scelto a seguito di vari tentativi in cerca del numero che restituisce le statistiche migliori

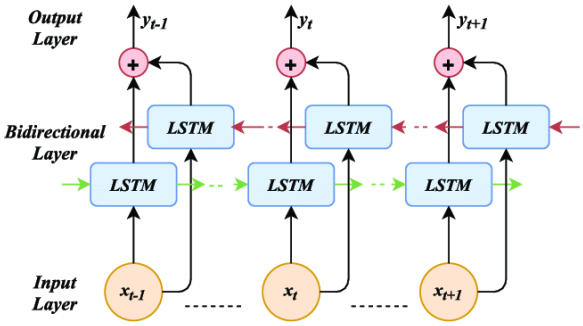


Figura 3.1: Modello di una rete LSTM bidirezionale che mostra i layer di ingresso e uscita. Le frecce rosse indicano percorso della sequenza a ritroso e quella verde in avanti.[19]

Bidirectional [18]

Serve a processare i dati in due direzioni opposte permettendo alla rete LSTM di considerare sia le informazioni passate che quelle future.

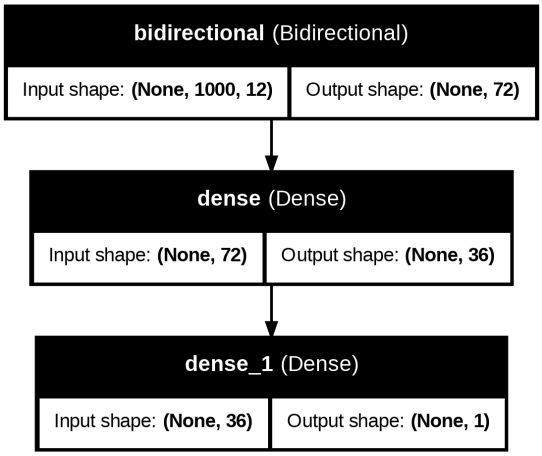
Dense [20]

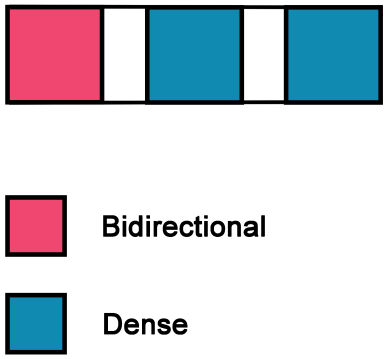
Serve a restituire un numero specifico di neuroni collegati ai precedenti. In questo caso si hanno 36 neuroni in uscita nel penultimo layer e 1 nell'ultimo essendo quello che restituisce il valore della classificazione. Inoltre, quest'ultimo layer è caratterizzato da una funzione di attivazione di tipo sigmoidea.

BatchNormalization [21]

Durante la progettazione si è provato anche ad aggiugnere un layer di BatchNormalization subito dopo l'input con lo scopo di normalizzare i valori ingresso.

```
model=Sequential()
model.add(Input(shape=(1000,12)))
model.add(BatchNormalization())
model.add(Bidirectional(LSTM(36)))
model.add(Dense(36))
model.add(Dense(1,activation='sigmoid'))
```

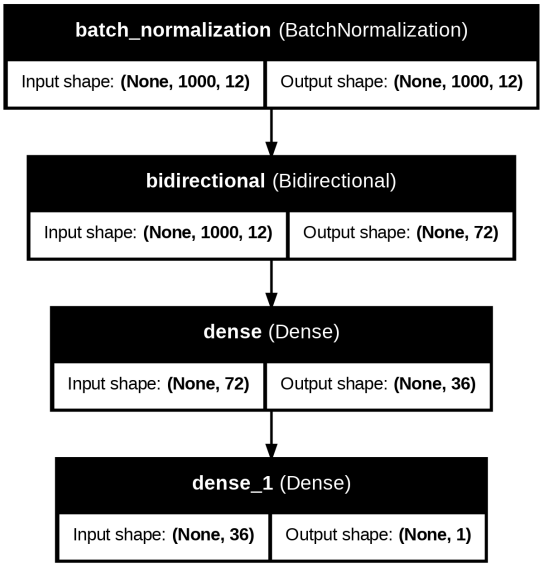


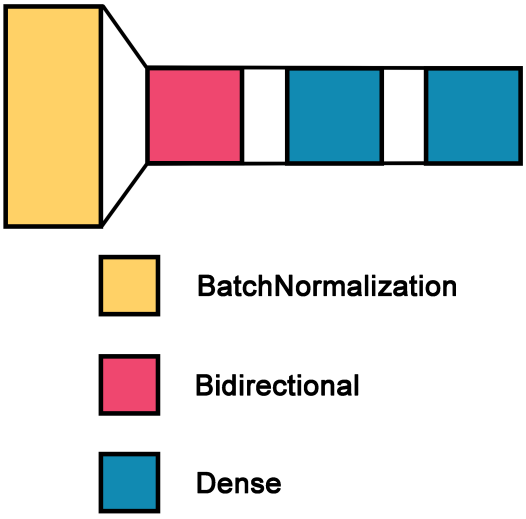


Layer (type)	Output Shape	Param #
bidirectional	(None,72)	14112
dense	(None,36)	2628
dense_1	(None,1)	37

	# Param
Total	16777
Trainable	16777
Non-trainable	0

Figura 3.2: Struttura del modello





Layer (type)	Output Shape	Param #
batch_normalization	(None, 1000, 12)	48
bidirectional	(None,72)	14112
dense	(None,36)	2628
dense_1	(None,1)	37

	# Param
Total	16825
Trainable	16801
Non-trainable	24

Figura 3.3: Struttura del modello con il layer di BatchNormalization

Algoritmo di ottimizzazione

Come algoritmo di ottimizzazione si è scelto Adam (Adaptive Moment Estimation)[22] con un passo di apprendimento di 10^{-3}.

```
model.compile(loss='binary_crossentropy',
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    metrics=["accuracy"])
```

3.2 Metriche utilizzate

Per valutare il modello costruito si sono usate come metriche l'accuratezza e la funzione di loss monitorandole sia sui dati di training che su quelli di validazione per poter valutare la presenza di overfitting, underfitting e, in generale, seguire l'andamento dell'addestramento.

Accuratezza

L'accuratezza rappresenta, in percentuale, quante classificazioni corrette sono state fatte rispetto al totale.
Nel caso in cui questo valore sia molto alto sui dati di training, ma basso in quelli di validazione, ci si ritrova nella situazione di overfitting, ovvero nel caso in cui il modello è troppo legato ai dati e non al loro significato, di conseguenza non è in grado di generalizzare le caratteristiche osservate e di riconoscerle su dei dati sconosciuti.

Loss

La funzione di perdita rappresenta la distanza tra il valore predetto dal modello e quello reale, con lo scopo di penalizzare gli errori.
In questo caso si è utilizzata una funzione di loss di tipo *binary crossentropy*.

3.3 Modalità di testing

Per testare il modello si sono utilizzati due metodi diversi:

1. Cross validation usando esclusivamente i dati privati;

2. Addestramento del modello sui dati pubblici e test su quelli privati.

3.3.1 Cross validation

É un metodo che consiste nel dividere un insieme di dati in k parti uguali utilizzandone una per il test e il resto per il training.
In questo caso si è scelta di utilizzare quella di tipo ”leave-one-out” che consiste nel utilizzare tutto il set tranne uno per il training e quello rimasto per il test, ripetendo il processo per l'intero dataset.
Per avere un riscontro sulla qualità del modello si calcola ad ogni ciclo la correttezza della previsione del test e la distanza tra il valore vero e quello predetto.

Per applicare questo metodo di validazione si è costruito un ciclo, iterato per tutti i segnali ECG, che:

1. preso l'insieme dei segnali privati, assegna l'ultimo al gruppo di test, mentre i restanti li usa per training;
2. addestra il modello sui dati di training utilizzando il 10% di questi come gruppo di validazione;
3. classifica il singolo segnale di test;
4. confronta il valore vero con quello predetto arrotondato e registra se la classificazione è corretta (1) o errata (0);
5. calcola la distanza tra il valore vero e quello predetto;
6. inserisce, al primo posto, il segnale di test tra quelli totali.

Finito l'intero ciclo, il codice restituisce l'accuratezza media dei test dell'intero processo, che viene calcolata tramite media aritmetica sui valori di correttezza (passaggio 4) e la loss media dei test, calcolata tramite media aritmetica delle distanze (passaggio 5).

```
X_test=(dati.pop())
Y_test=(diagn.pop())
X_train=np.array(dati)
Y_train=np.array(diagn)
```

Estrazione dati di test (punto 1) dove la variabile *dati* è l'insieme dei segnali ECG mentre *diagn* quello delle diagnosi

```
p=model.predict(np.reshape(np.array(X_test),(1,10000,12)))
Prev = round(p[0][0])

if Prev==Y_test:
    acc_per_fold.append(1)                # giusto
else:
    acc_per_fold.append(0)                # sbagliato

loss_per_fold.append(np.abs(Y_test-p))

dati.insert(0,X_test)
diagn.insert(0,Y_test)
```

Predizione della diagnosi del test, valutazione della sua correttezza, calcolo della distanza tra il valore vero e quello predetto, e reinserimento del dato (punti 4, 5, 6)

3.3.2 Addestramento sul dataset pubblico

Per avere più dati da dedicare all'addestramento, si è scelto di utilizzare il database pubblico per il training e quello privato esclusivamente per il test, in modo da verificare che il modello, anche se addestrato con dei dati di origine diversa, sia adeguato per la classifica degli ECG privati.
Per poter avere dei segnali comparabili, i segnali privati sono stati sottocampionati a 100Hz come descritto in 3.1.2.

Capitolo 4

Risultati

In questo capitolo si presentano i risultati del modello, descritto in 3, addestrato sui dati privati nella sezione 4.1 e sui dati pubblici nella sezione 4.2, entrambi poi testati sul database privato.

4.1 Database privato

Utilizzanto la cross validation, come descritto in 3.3.1, il modello è stato addestrato 109 volte, ovvero il numero di segnali del dataset privato, e poi testato su tutti i campioni, uno per volta. Visto che ogni modello addestrato ha risultati diversi, i quali però sono equiparabili tra loro, si sono scelti due cicli con valori rappresentativi dei 109 addestramenti: per il primo modello, quello senza il layer di BatchNormalization, si è scelto il 96° ciclo (tabella 4.1) e per il secondo modello, quello con il layer di BatchNormalization, l'83° (tabella 4.2).

4.1.1 Modello senza il layer di BatchNormalization

I risultati ottenuti sui test sono:

- Accuracy media = 0.4679
- Loss media = 0.5106

4.1.2 Modello con il layer di BatchNormalization

I risultati ottenuti sui test sono:

- Accuracy media = 0.4404
- Loss media = 0.5191

Epoca	Accuracy	Loss	Validation Accuracy	Validation Loss
1	0.4714	0.6962	0.4545	0.7229
2	0.4886	0.6911	0.5455	0.7115
3	0.5404	0.6893	0.4545	0.7049
4	0.5396	0.6852	0.4545	0.7010
5	0.5819	0.6786	0.4545	0.6937
6	0.6270	0.6726	0.4545	0.6904
7	0.6291	0.6606	0.4545	0.6888
8	0.6319	0.6500	0.3636	0.6839
9	0.6693	0.6262	0.6364	0.6519
10	0.6506	0.6039	0.4545	0.7536
11	0.7483	0.5761	0.4545	0.8399
12	0.6968	0.5775	0.3636	0.8414
13	0.7502	0.5112	0.2727	0.7989
14	0.6996	0.5444	0.3636	0.8051
15	0.7778	0.4864	0.4545	0.9672

Tabella 4.1: Risultati 96° ciclo del primo modello - dataset privato

Epoca	Accuracy	Loss	Validation Accuracy	Validation Loss
1	0.5972	0.6963	0.4545	0.6966
2	0.6380	0.6783	0.2727	0.7030
3	0.6983	0.6421	0.2727	0.7092
4	0.6685	0.6347	0.4545	0.7163
5	0.7108	0.6048	0.4545	0.7256
6	0.6566	0.6057	0.4545	0.7357
7	0.7156	0.5608	0.4545	0.7509
8	0.6983	0.5567	0.4545	0.7625
9	0.7661	0.5417	0.4545	0.7649
10	0.7238	0.5249	0.5455	0.7631
11	0.7404	0.5045	0.3636	0.7614
12	0.7834	0.4533	0.3636	0.7806
13	0.8035	0.4405	0.4545	0.7918
14	0.8637	0.3959	0.5455	0.8015
15	0.8478	0.3395	0.6364	0.8202

Tabella 4.2: Risultati 83° ciclo del secondo modello - dataset privato

4.2 Database pubblico

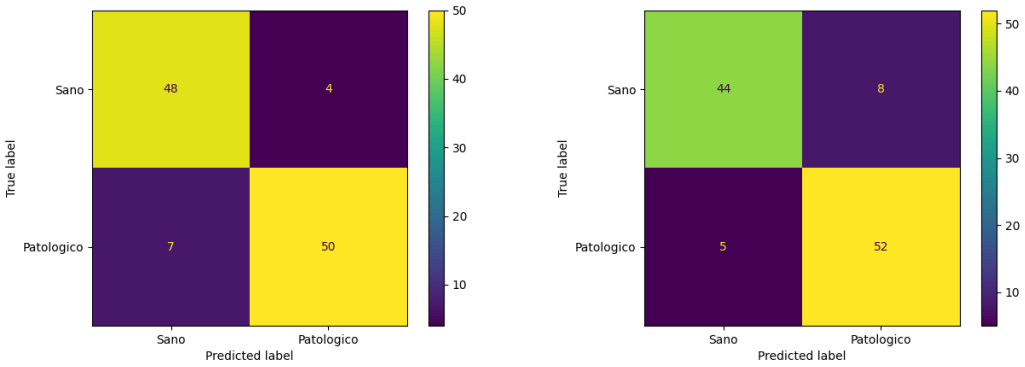
L'addestramento del modello sui dati pubblici, su 30 epoche, ha restituito i risultati esposti nelle tabelle 4.3 e 4.4.

4.2.1 Modello senza il layer di BatchNormalization

Questo modello, testato sui dati privati, ha restituito una accuratezza dell'89.91%. In particolare, come illustrato dalla matrice di confusione 4.1a, su 109 segnali classificati, ha predetto correttamente 98 segnali ECG ed erroneamente 11, di cui 4 sono falsi patologici e 7 falsi sani.

4.2.2 Modello con il layer di BatchNormalization

Questo modello, testato sui dati privati, ha restituito una accuratezza dell'88.07%. In particolare, come illustrato nella matrice di confusione 4.1b, su 109 segnali classificati, ha predetto correttamente 96 segnali ECG ed erroneamente 13, di cui 8 falsi patologici e 5 falsi sani.



(a) Primo modello

(b) Secondo modello

Figura 4.1: Matrici di confusione

Epoca	Accuracy	Loss	Validation Accuracy	Validation Loss
1	0.5560	0.6792	0.5889	0.6428
2	0.6159	0.6384	0.5726	0.6509
3	0.6148	0.6296	0.5942	0.6119
4	0.6628	0.5933	0.6335	0.6584
5	0.6046	0.6315	0.7345	0.5113
6	0.7520	0.5022	0.8253	0.3847
7	0.8211	0.4025	0.8462	0.3583
8	0.8452	0.3585	0.8559	0.3261
9	0.8572	0.3342	0.8666	0.3007
10	0.8649	0.3194	0.8618	0.3056
11	0.8706	0.3079	0.8659	0.2994
12	0.8733	0.3000	0.8630	0.2961
13	0.8767	0.2980	0.8730	0.2879
14	0.8775	0.2866	0.8699	0.3005
15	0.8838	0.2798	0.8710	0.2970
16	0.8823	0.2765	0.8720	0.2829
17	0.8837	0.2793	0.8750	0.2785
18	0.8845	0.2765	0.8755	0.2784
19	0.8856	0.2646	0.8768	0.2775
20	0.8744	0.2836	0.8559	0.3285
21	0.8700	0.3091	0.8745	0.2877
22	0.8838	0.2700	0.7332	0.5122
23	0.8379	0.3532	0.8725	0.2841
24	0.8936	0.2582	0.8722	0.2832
25	0.8917	0.2627	0.8758	0.2844
26	0.8902	0.2574	0.8766	0.2798
27	0.8909	0.2533	0.8776	0.2752
28	0.8939	0.2521	0.8712	0.2993
29	0.8958	0.2479	0.8761	0.2815
30	0.8960	0.2449	0.8743	0.2869

Tabella 4.3: Risultati del primo modello - dataset pubblico

Epoca	Accuracy	Loss	Validation Accuracy	Validation Loss
1	0.6319	0.6348	0.7276	0.5259
2	0.7095	0.5597	0.7794	0.4499
3	0.7713	0.4757	0.8189	0.4013
4	0.8104	0.4165	0.8350	0.3690
5	0.8295	0.3787	0.8393	0.3562
6	0.8318	0.3726	0.8437	0.3428
7	0.8419	0.3529	0.8276	0.3842
8	0.8475	0.3480	0.8511	0.3267
9	0.8528	0.3332	0.8551	0.3238
10	0.8514	0.3267	0.8511	0.3317
11	0.8596	0.3207	0.8587	0.3250
12	0.8667	0.3162	0.8610	0.3131
13	0.8659	0.3081	0.8605	0.3219
14	0.8728	0.2962	0.8638	0.3152
15	0.8717	0.2963	0.8528	0.3541
16	0.8730	0.2958	0.8704	0.3119
17	0.8757	0.2855	0.8666	0.3121
18	0.8815	0.2772	0.8625	0.3227
19	0.8805	0.2842	0.8694	0.3086
20	0.8835	0.2701	0.8669	0.3070
21	0.8860	0.2640	0.8590	0.3137
22	0.8892	0.2598	0.8613	0.3317
23	0.8902	0.2573	0.8697	0.3084
24	0.8964	0.2506	0.8669	0.3120
25	0.8960	0.2462	0.8610	0.3149
26	0.8985	0.2416	0.8587	0.3257
27	0.8989	0.2449	0.8541	0.3304
28	0.9011	0.2354	0.8212	0.3804
29	0.8894	0.2605	0.8498	0.3469
30	0.8997	0.2393	0.8613	0.3185

Tabella 4.4: Risultati del secondo modello - dataset pubblico

Capitolo 5

Discussione dei risultati

5.1 Database privato

Come si può notare dal grafico 5.1, l'accuratezza dell'addestramento migliora di epoca in epoca mentre quella della validazione ha delle prestazioni molto inferiori, come si può osservare anche dalla funzione di loss che, nel caso del training, diminuisce ad ogni epoca, mentre nella validazione diverge crescendo, illustrato nel grafico 5.2.
Con l'aggiunta del layer di BatchNormalization la tendenza è la stessa ma procede più regolarmente (grafici 5.3 e 5.4).
Questo accade perché il database privato è troppo piccolo per poter costituire un buon insieme di addestramento, di conseguenza il modello si adatta troppo fedelmente ai dati e non è in grado di classificare correttamente i segnali sconosciuti (il gruppo di validazione).

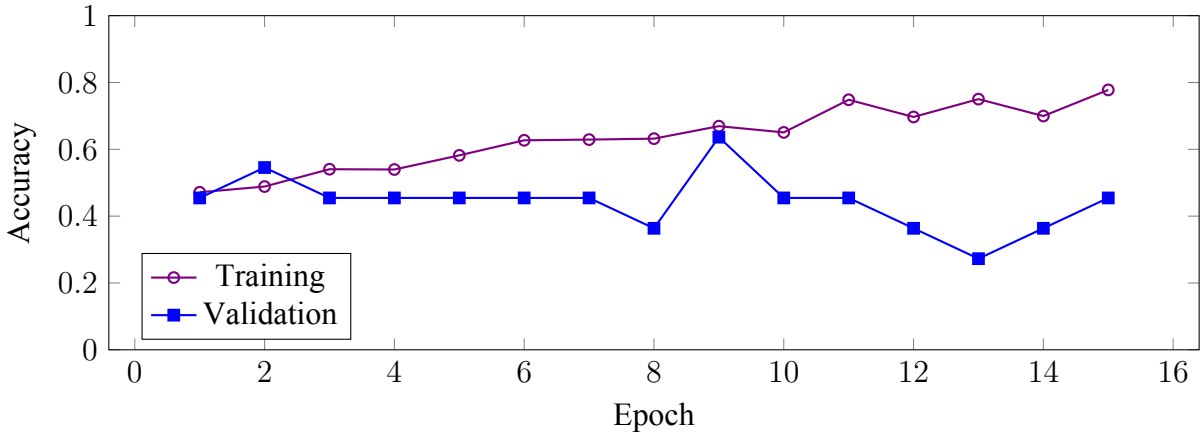


Figura 5.1: Accuratezza 96° ciclo del primo modello 4.1 - dataset privato

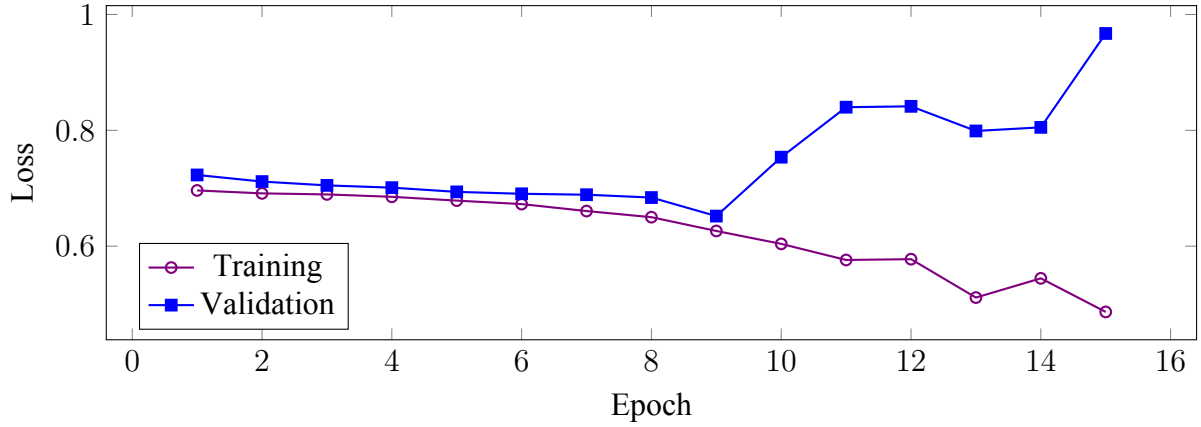


Figura 5.2: Loss 96° ciclo del primo modello 4.1 - dataset privato

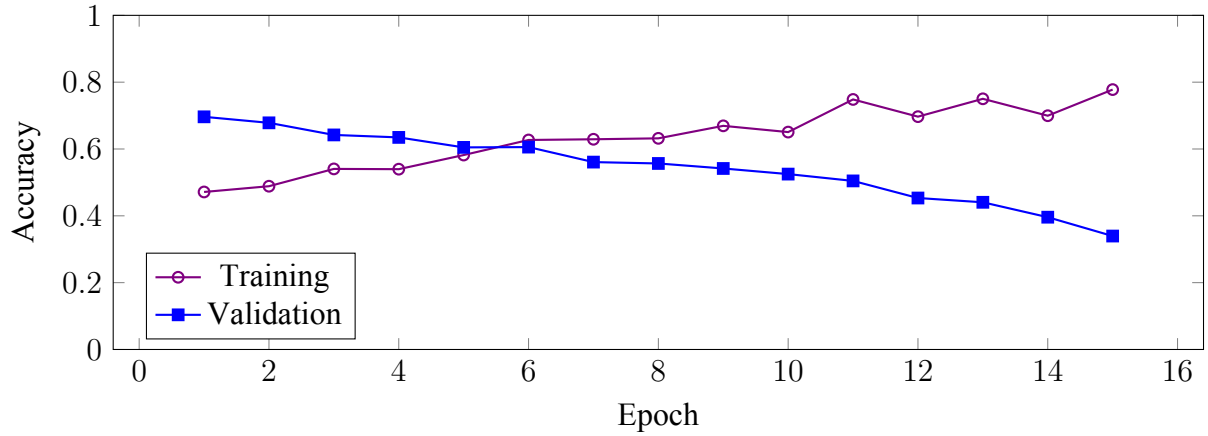


Figura 5.3: Accuratezza 83° ciclo del secondo modello 4.2 - dataset privato

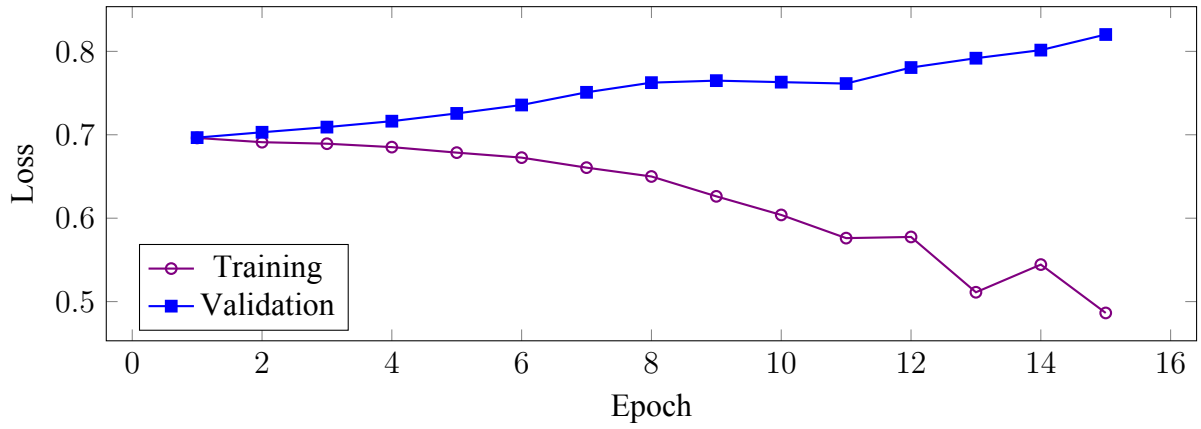


Figura 5.4: Loss 83° ciclo del secondo modello 4.2 - dataset privato

5.2 Database pubblico

I risultati ottenuti con il database pubblico sono più positivi, infatti, come si può notare dai grafici di training e validazione, l'accuratezza (grafici 5.5 e 5.7) cresce rapidamente e resta stabile attorno all'80%, e la funzione di loss (grafici 5.6 e 5.8) decresce stabilizzandosi attorno allo 0.3. Questi risultati si riscontrano anche nel test sui dati privati, ottenendo valori di accuratezza dell'89.91% nel primo modello (4.1a) e dell'88.07% nel secondo (4.1b).
Pur avendo risultati simili, si può osservare una tendenza del primo modello a classificare, nel caso di errore, più falsi sani che falsi patologici, mentre nel secondo la tendenza è opposta[1] (come si può osservare dalle matrici di confusione 4.1).

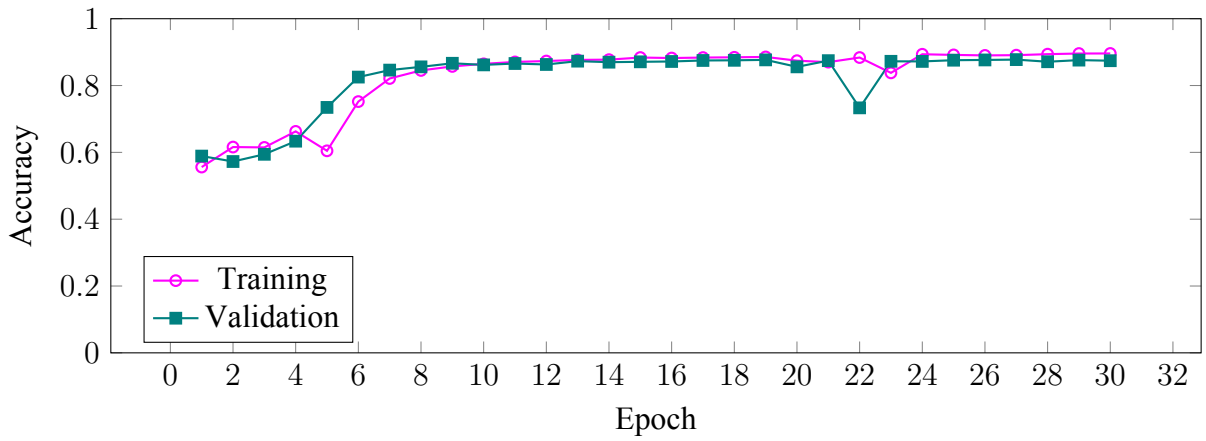


Figura 5.5: Accuratezza sul dataset pubblico 4.3

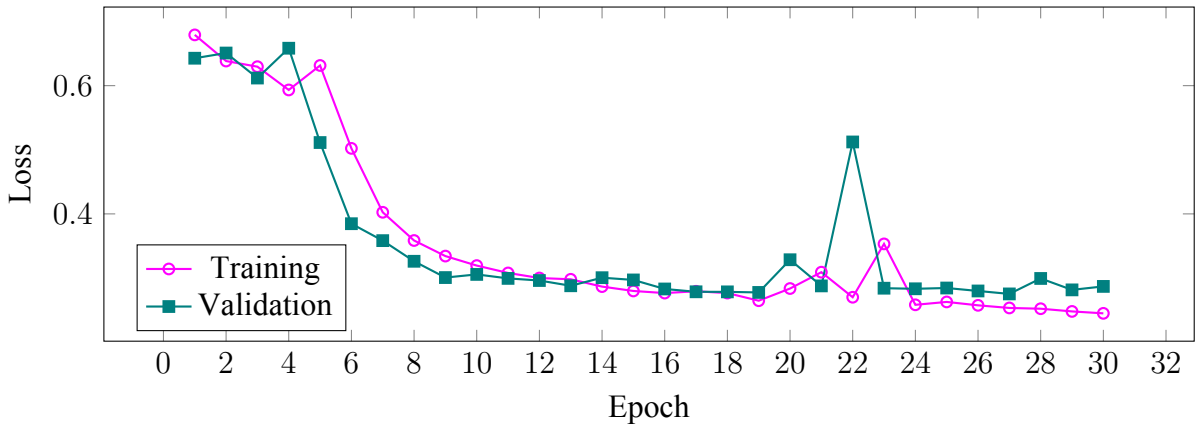


Figura 5.6: Loss sul dataset pubblico 4.3

[1]Questa propensione si è osservata anche durante gli altri addestramenti del modello usando valori di batch e di epoche diversi

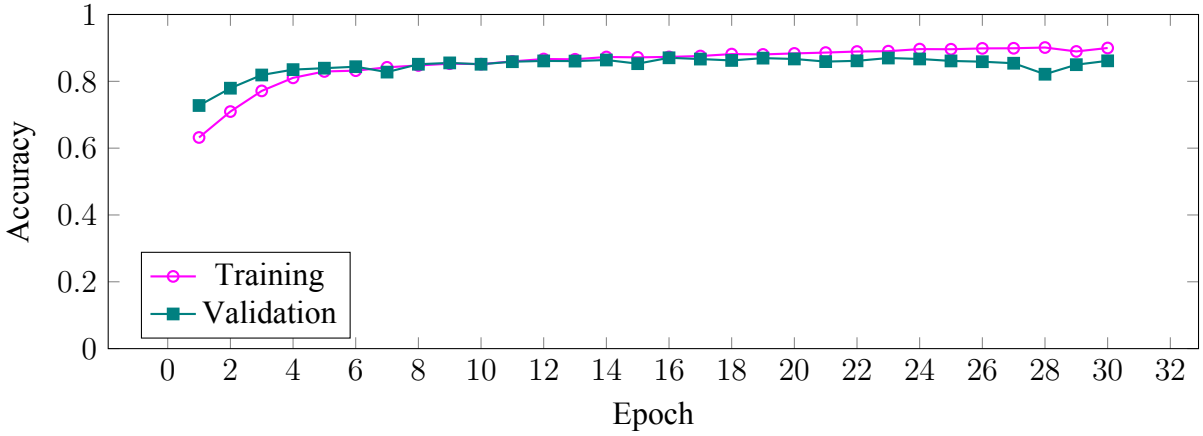


Figura 5.7: Accuratezza sul dataset pubblico con il layer di BatchNormalization 4.4

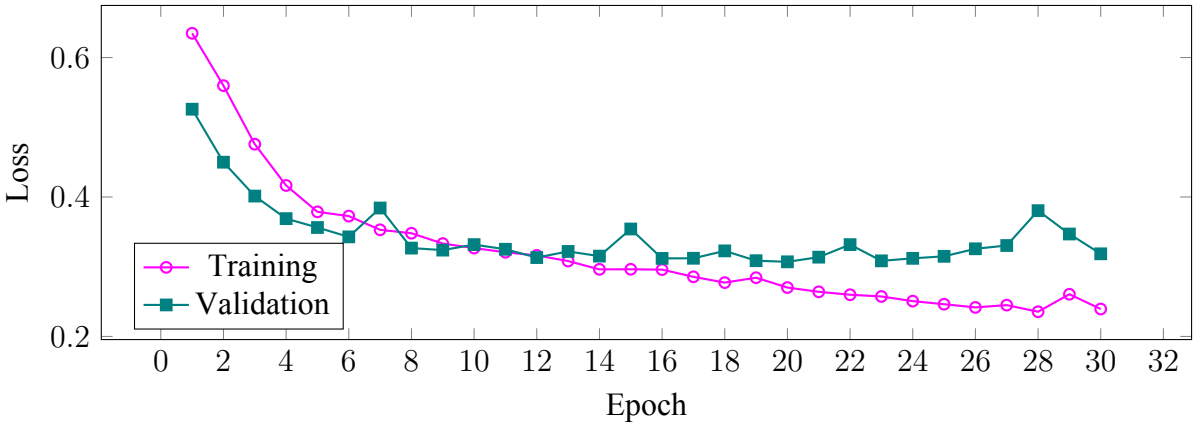


Figura 5.8: Loss sul dataset pubblico con il layer di BatchNormalization 4.4

Capitolo 6

Conclusione

Alla luce dei risultati ottenuti dai diversi modelli e metodi utilizzati, si deduce che i modelli addestrati sul database pubblico hanno valori di accuratezza migliori anche quando testati sui segnali ECG privati.
Inoltre, sembrerebbe che il layer di BatchNormalization possa ridurre il numero di segnali patologici classificati come sani (falsi negativi) e, di conseguenza, sarebbe più adatto ad una applicazione clinica, anche se la diagnosi finale spetta sempre al medico.
In conclusione, nonostante i pochi dati privati a disposizione, in questo lavoro di tesi è stata sviluppata una pipeline generica (per l'addestramento e la validazione della rete LSTM) che può essere utilizzata quando ci sarà una quantità più ampia di dati a disposizione.

6.1 Difficoltà riscontrate

Durante lo sviluppo di questo progetto si sono incontrate varie difficoltà e problemi relativi sopratutto ai dati privati e ai lunghi tempi di esecuzione.
Nello specifico, la prima di queste è la carenza dei dati privati che ha reso impossibile un addestramento esclusivo su di essi che portasse ad un buon modello.
Inoltre, i dati privati non hanno una diagnosi verificata (*ground truth*), di conseguenza vi è un margine d'errore non trascurabile sui valori di accuratezza dei test sia per l'addestramento che per la verifica delle classi.
É a causa di questi motivi che ci si è dovuti rivolgere al database pubblico che, però, ha portato ad ulteriori due limitazioni: i dispositivi di acquisizione dei segnali ECG dei due dataset sono diversi, così come la frequenza di campionamento. Quest'ultima, in particolare, ha comportato un sottocampionamento dei segnali privati e quindi una perdita di informazioni.
Infine, un limite di questo modello è che la rete neurale su cui si basa, la LSTM, ha dei tempi

di addestramento molto lunghi, soprattutto in combinazione con la bidirezionalità, che impiega dunque molte risorse.

6.2 Aspetti futuri

Per poter avere delle metriche più accurate è necessario, quindi, addestrare il modello su un database privato molto più vasto e con delle diagnosi verificate da almeno un medico.
Per il futuro, inoltre, sarebbe interessante ampliare il modello in modo che tenga in considerazione anche altri parametri personali, come età e sesso, per classificare i segnali ECG nelle due classi, ma anche per riconoscere le patologie specifiche.

Bibliografia

[1] Indirizzo: https://www.m31.com/.

[2] Indirizzo: https://litfl.com/normal-sinus-rhythm-ecg-library/.

[3] Indirizzo: https://oxfordmedicaleducation.com/ecgs/ecg-examples/.

[4] J. G. Webster e J. W. Clark, *Medical Instrumentation: Application and Design - 4th ed*. John Wiley & Sons Inc, 2009.

[5] Indirizzo: https://www.amboss.com/us/knowledge/ecg.

[6] A. H. Kashou, A. M. May e P. A. Noseworthy, «Artificial Intelligence-Enabled ECG: a Modern Lens on an Old Technology,» *Current Cardiology Reports*, 2020. DOI: 10.1007/s11886-020-01317-x.

[7] Z. Ebrahimi, M. Loni, M. Daneshtalab e A. Gharehbaghi, «A review on deep learning methods for ECG arrhythmia classification,» *Elsevier*, 2020.

[8] S. Chauhan e L. Vig, «Anomaly Detection in ECG Time signals via Deep Long Short-Term Memory Networks,» *IEEE*, 2015.

[9] S. Hochreiter e J. Schmidhuber, «Long Short-Term Memory,» *Neural Computation*, vol. 9, n. 8, pp. 1735–1780, nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[10] S. Saadatnejad, M. Oveisi e M. Hashemi, «LSTM-Based ECG Classification for Continuous Monitoring on Personal Wearable Devices,» *IEEE journal of biomedical and health informatics*, 2020.

[11] P. Wagner, N. Strodthoff, R. Bousseljot, W. Samek e T. Schaeffter, «PTB-XL, a large publicly available electrocardiography dataset (version 1.0.3),» *PhysioNet*, 2022. indirizzo: https://doi.org/10.13026/kfzx-aw45.

[12] *Elettrocardiografo Mortara WAM-PC*. indirizzo: https://www.e-htn.it/device/elettrocardiografo-mortara-wam-pc.

[13] Innolitics, *DICOM Standard Browser*. indirizzo: https://dicom.innolitics.com/ciods/12-lead-ecg/waveform-annotation/0040b020.

[14] *Classify ECG Signals Using Long Short-Term Memory Networks*. indirizzo: https://it.mathworks.com/help/signal/ug/classify-ecg-signals-using-long-short-term-memory-networks.html.

[15] *Documentazione Sequential TensorFlow*. indirizzo: https://www.tensorflow.org/api_docs/python/tf/keras/Sequential.

[16] *Documentazione Input TensorFlow*. indirizzo: https://www.tensorflow.org/api_docs/python/tf/keras/Input.

[17] *Documentazione LSTM TensorFlow*. indirizzo: https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM.

[18] *Documentazione Bidirectional TensorFlow*. indirizzo: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Bidirectional.

[19] I. K. Ihianle, A. O. Nwajana, S. H. Ebenuwa, R. I. Otuka, K. Owa e M. O. Orisatoki, «A Deep Learning Approach for Human Activities Recognition From Multimodal Sensing Devices,» *IEEE Access*, vol. 8, pp. 179 028–179 038, 2020. DOI: 10.1109/ACCESS.2020.3027979.

[20] *Documentazione Dense TensorFlow*. indirizzo: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense.

[21] *Documentazione BatchNormalization TensorFlow*. indirizzo: https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization.

[22] *Documentazione Adam TensorFlow*. indirizzo: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam.