Epilepsy Alert

Documentazione Progetto

Gruppo di Lavoro:

Nicola Lieggi, 699184, n.lieggi2@studenti.uniba.it

Link Repository GitHub:

EpilepsyAlert

AA 2023-2024

Indice

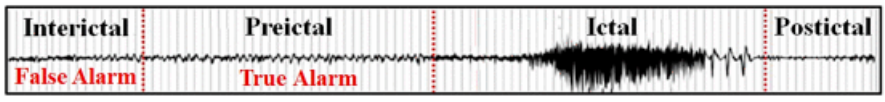
Capitolo 1

Introduzione

L'epilessia è una delle malattie neurologiche più diffuse al mondo con circa 50 milioni di persone che ne soffrono, l'80% delle quali vive in paesi a basso e medio reddito. È caratterizzata da crisi ricorrenti che possono avvenire anche più volte al giorno, brevi episodi di movimento involontario che coinvolgono una o più parti del corpo, e perdita di coscienza. Chi ne soffre, oltre a rischiare problemi fisici come fratture e lividi, tende ad avere anche problemi di natura psicologica come ansia e depressione. Questo porta spesso ad isolarsi dalla società, peggiorando ulteriormente la situazione psicologica di chi vive con questa condizione. Inoltre, in alcune parti del mondo, persiste ancora uno stigma nei confronti di chi soffre di tale malattia, influenzando negativamente non solo la loro vita, ma anche quella dei loro familiari.

Nella maggior parte dei casi l'epilessia può essere gestita con farmaci antiepilettici. Tuttavia in alcune situazioni questi non sono sufficienti e, inoltre, nei paesi più poveri, l'accesso a tali farmaci può essere estremamente difficile, rendendo necessarie altre misure per prevenire le crisi. Un'alternativa è quella di analizzare l'elettroencefalogramma (EEG) al fine di individuare pattern anomali che indicano che una crisi sta per accadere [1]. L'EEG è la registrazione del segnale elettrico dell'attività celebrale, è considerato il miglior strumento diagnostico per l'epilessia per diversi motivi: innanzitutto è poco invadente, infatti consiste di una serie di elettrodi posizionati sul cuoio capelluto che permettono di rilevare il segnale elettrico del cervello, quindi non implica dolore ed è considerato un esame sicuro. L'analisi dell'elettroencefalogramma divide il segnale elettrico in fasi che possono essere di 4 tipologie[2]:

- interictali: indicano il periodo temporale fra due crisi;
- preictali: periodo poco prima dell'arrivo di una crisi;
- ictali: periodo durante la crisi;
- postictali: periodo poco dopo la crisi.



La previsione delle crisi attraverso la rilevazione delle fasi preictali consentirebbe di allertare un soggetto che soffre di epilessia in tempo per potersi recare in ospedale o poter chiedere aiuto a qualcuno. Un sistema del genere migliorerebbe di molto la qualità della vita di queste persone. L'analisi dell'EEG prevede la presenza di una figura professionale formata per controllare il segnale e rilevare l'arrivo di una crisi, figura che per ovvie ragioni non può essere sempre presente con il soggetto che soffre di epilessia. Per questo motivo si è immaginato un dispositivo portatile, come per esempio un cappello o un berretto, dotato di elettrodi che monitorano il soggetto 24/24h registrando l'EEG e inviando ogni x minuti (o secondi) un segmento di registrazione di qualche secondo a un dispositivo sul

quale è presente un modello in grado di predire se il segmento ricevuto rientra nella fase preictale e, eventualmente, avvisare della possibile insorgenza di una crisi. Obbiettivo di questo progetto è proprio la creazione del modello sopra citato, modello che deve essere personalizzato perché le caratteristiche delle crisi possono variare da soggetto a soggetto e per garantire la massima affidabilità deve essere allenato su dati propri di uno specifico paziente.

Capitolo 2

Dataset

2.1 Database

Per la creazione del dataset è stato utilizzato un database di EEG dell'ospedale pediatrico di Boston [3]. Consiste di registrazioni EEG appartenenti a 22 soggetti che soffrono di crisi epilettiche intrattabili, ovvero che non risponde adeguatamente ai trattamenti farmacologici standard. Le registrazioni sono state raggruppate in 23 gruppi, uno per ogni paziente. Tutte le informazioni che consentirebbero di identificare un soggetto sono state adeguatamente oscurate o modificate con dati fittizi dai creatori del database. Ciascun caso contiene fra le 9 e le 42 registrazioni in formato .edf. Date le limitazioni hardware le registrazioni non sono sempre continue, in alcuni casi abbiamo gap fra una registrazione e l'altra di una manciata di secondi, mentre in altri casi superano anche l'ora. Ogni registrazione può avere un numero di canali che varia da 23 a 26 e tutti i segnali sono stati campionati a 256 Hz.

In particolare si è considerato il caso `chb01`, dato che tutte le registrazione utilizzano gli stessi canali e che è stato possibile estrarre ben 10 ore totali di EGG ininterrotti fra fasi interictali e preictali.

Paziente	Sesso	Età	No. crisi	Durata registrazioni
chb01	F	11	7	40:33:08

2.2 Creazione del Dataset

Per la creazione del dataset sono stati seguiti due approcci: il primo consiste nell'estrazione dei segnali EEG di interesse dai file `.edf`, e il secondo che consiste nell'elaborazione dei segnali estratti in metriche, ottenendo quindi due dataset che descrivono gli stessi segnali da due punti di vista differenti. Questa operazione è stata effettuata per via dell'elevata complessità dei risultati del primo approccio, dato che per ogni secondo di registrazione abbiamo 256 campioni per 23 canali.

2.2.1 Primo Approccio

Nel primo approccio sono stati utilizzati i segnali EEG grezzi, senza che nessun preprocessing sia stato applicato ad essi. Sono state estratte dai file `edf` due delle quattro fasi: interictali e preictali, rendendo di fatto il problema un problema di classificazione binaria dove le etichette sono:

- 0: per le fasi interictali;
- 1: per le fasi preictali.

Un aspetto importante relativo alle fasi menzionate è la durata. Le caratteristiche delle crisi possono variare significativamente da soggetto a soggetto, motivo per cui non esistono in letteratura scientifica delle durate standard per ciascuna fase. Per esigenze di semplificazione, sono state considerate durate indicative di minimo quattro ore per le fasi interictali e di un'ora fissa per quelle preictali. Non è escluso che andando a modificare questi parametri si possa ottenere una rappresentazione migliore di tali fasi.

La funzione `make_dataset` (presente all'interno di `preprocess.py`), data l'ora di inizio di una crisi, estrae un totale di 5 ore di dati relativi alla fase interictale e preictale. Tuttavia, alcune crisi avvengono a meno di cinque ore dalla precedente, in alcuni casi passano soltanto pochi minuti, falsando quindi i risultati dell'estrazione. Di conseguenza `make_dataset` estrae soltanto le crisi principali, ovvero le crisi che nelle cinque ore precedenti dall'inizio della fase ictale non hanno crisi sovrapposte. Avendo una durata fissa, ed essendo le registrazioni non necessariamente continue, dalla fase preictale non è sempre possibile estrarre esattamente un ora di dati. Problema che non si verifica con le fasi interictali che avendo una durata minima di quattro ore, anche in presenza di interruzioni, permettono comunque di ottenere l'intero intervallo di dati. Un altro aspetto importante è quello relativo alla segmentazione. Infatti una volta individuate le crisi principali ed estratti i segnali di interesse, questi devono essere divisi in segmenti. In questo caso si è scelto di dividere il segnale in segmenti da 5 secondi, anche questo aspetto è tuttavia soggetto alle caratteristiche personali dei pazienti e quindi è un parametro che in futuro potrebbe essere determinato più precisamente. La funzione `make_dataset`, consente anche il bilanciamento in modo da avere 50% di segmenti interictali e 50% preictali e di dividere il dataset in training e test set (80% training e 20% test). Il risultato è un dataset contenente serie temporali di cinque secondi, dove ciascuna serie può essere interictale o preictale. Una serie di cinque secondi è rappresentata da array bidimensionale di $1280 * 23$ valori, dove 1280 è il numero di valori per un canale ($256 * 5$, dove 256 è la frequenza di campionamento al secondo) e 23 è il numero di canali.

2.2.2 Secondo Approccio

Nel secondo approccio si è deciso di elaborare ulteriormente il dataset per semplificarlo ed ottenere una rappresentazione bidimensionale (invece che tridimensionale), decisamente più semplice da gestire. Dato un elemento del dataset, per ogni canale sono state estratte cinque *feature*:

- valore medio;
- valore efficace (rms, *root mean square*);
- deviazione standard;
- curtosi;
- indice di asimmetria (skewness).

Come risultato si è ottenuto un dataset composto da $23 * 5 = 115$ *feature*. Per realizzarlo è stata impiegata la libreria TSFEL. La decisione di estrarre queste *feature* è derivata dai risultati dell'articolo *Trends in eeg signal feature extraction applications* [4]. Il tutto è stato implementato nella funzione `preprocess_dataset` che, dato il dataset risultante dall'approccio precedente, restituisce un nuovo dataset aventi le caratteristiche appena descritte.

Capitolo 3

Knowledge Base

Di seguito si riporta come è stata realizzata la Base di Conoscenza, a partire dall'individuazione degli individui fino alla definizione delle clausole definite. La KB è stata realizzata usando il linguaggio Prolog.

3.1 Individui

La rappresentazione del dominio di interesse è iniziata con l'individuazione degli individui. In particolare sono stati individuate le seguenti due classi di individui, rappresentati attraverso i rispettivi simboli di funzione:

- `patient(P_ID)`: paziente identificato con `P_ID`;
- `eegRec(R_ID)`: registrazione EEG identificata con `R_ID`;
- `seizure(S_ID)`: crisi epilettica identificata con `S_ID`.

3.2 Relazioni

Sono state individuate le seguenti due relazioni, la prima collega i pazienti alle registrazioni e la seconda le registrazioni alle crisi epilettiche:

- `hasRecording(P_ID, R_ID)`: vero se al paziente `P_ID` è associata una registrazione EEG `R_ID`.
- `recordsSeizure(R_ID, S_ID)`: vero se la registrazione `R_ID` ha registrato la crisi `S_ID`.

3.3 Proprietà

Non sono state incluse tutte le proprietà perché alcune non sono state reputate utili ad inferire nuove informazioni. Per esempio, nel *dataset* originario ai pazienti sono associate informazioni sul sesso e sull'età che non hanno nessuna utilità in questo caso.

3.3.1 Proprietà per eegRec

Per le registrazioni EEG sono disponibili le seguenti proprietà:

- `recTimes(R_ID, R_start, R_end)`: orari di inizio e fine della registrazione `R_ID`;

- `channels(R_ID, Channel)`: canale per la registrazione `R_ID`;
- `samplingRate(R_ID, Rate)`: frequenza di campionamento per la registrazione `R_ID`.

3.3.2 Proprietà per seizure

Associate a ciascuna crisi epilettica abbiamo le seguenti proprietà:

- `seizureTimes(S_ID, S_start, S_end)`: secondi di inizio e fine dall'inizio della registrazione nella quale la crisi `S` è stata rilevata;

3.4 Clausole Definite

Di seguito si descrivono alcune clausole definite fondamentali per ragionare sulla KB:

- recupera tutte le registrazioni EEG di un paziente:

```
patientRecordings(P_ID, R_ID) :-
  patient(P_ID),
  hasRecording(P_ID, R_ID)
```

- recupera le crisi epilettiche di un paziente:

```
patientSeizures(P_ID, S_ID) :-
  patient(P_ID),
  hasRecording(P_ID, R_ID),
  recordsSeizure(R_ID, S_ID)
```

- data una crisi, restituire gli orari di inizio e fine della fase preictale:

```
preictal(S_ID, Preictal_start, Preictal_end) :-
  seizureTimes(S_ID, S_start, S_end),
  Preictal_end is S_start,
  Preictal_start is S_start - 3600
```

- data una crisi, restituire gli orari di inizio e fine della fase interictale:

```
interictal(S_ID, Interictal_start, Interictal_end) :-
  seizureTimes(S_ID, S_start, S_end),
  Preictal_start is S_start - 3600,
  Interictal_end is Preictal_start,
  Interictal_start is Preictal_start - 14400
```

- data una registrazione EEG e un ora di inizio, restituire un segmento di 5 secondi:

```
segment(R_ID, T_start, T_end) :-
  recTimes(R_ID, R_start, R_end),
  T_start >= R_start,
  T_end =< R_end,
  T_end is T_start + 5
```

- dato un orario di inizio e di fine, restituisce le crisi che si verificano in questo intervallo:

```
seizure_in_time_window(Start_time, End_time, S_ID) :-
  seizure(S_ID),
  seizureTimes(S_ID, S_start, S_end),
  S_start =< End_time,
  S_end >= Start_time
```

Di seguito invece, vengono riportate le clausole per la creazione del dataset definito in 2.2.2. Per l'estrazione dei dati si assume l'esistenza di una clausola ipotetica `extract_data_segment` che estrae i dati per una registrazione, un canale e un segmento di 5 secondi.

- clausola per calcolare la media su un canale per un segmento specifico di 5 secondi:

```
mean_segment(R_ID, Channel, T_start, T_end, Mean) :-
  recTimes(R_ID, R_start, R_end),
  T_start >= R_start,
  T_end =< R_end,
  T_end is T_start + 5,
  extract_data_segment(R_ID, Channel, T_start, T_end, Data),
  mean(Data, Mean)
```

- calcolare il root mean square:

```
rms_segment(R_ID, Channel, T_start, T_end, RMS) :-
  recTimes(R_ID, R_start, R_end),
  T_start >= R_start,
  T_end =< R_end,
  T_end is T_start + 5,
  extract_data_segment(R_ID, Channel, T_start, T_end, Data),
  root_mean_square(Data, RMS)
```

- calcolare la deviazione standard:

```
stddev(R_ID, Channel, T_start, T_end, StdDev) :-
  recTimes(R_ID, R_start, R_end),
  T_start >= R_start,
  T_end =< R_end,
  T_end is T_start + 5,
  extract_data_segment(R_ID, Channel, T_start, T_end, Data),
  standard_deviation(Data, StdDev)
```

- calcolare la curtosi:

```
kurtosis(R_ID, Channel, T_start, T_end, Kurtosis) :-
  recTimes(R_ID, R_start, R_end),
  T_start >= R_start,
  T_end =< R_end,
  T_end is T_start + 5,
  extract_data_segment(R_ID, Channel, T_start, T_end, Data),
  kurtosis(Data, Kurtosis)
```

- calcolare l'indice di asimmetria:

```
skewness(R_ID, Channel, T_start, T_end, Skewness) :-
  recTimes(R_ID, R_start, R_end),
  T_start >= R_start,
  T_end =< R_end,
  T_end is T_start + 5,
  extract_data_segment(R_ID, Channel, T_start, T_end, Data),
  skewness(Data, Skewness)
```

Capitolo 4

Modelli

4.1 Introduzione

Di seguito si riportano i modelli con il quale si è riusciti a raggiungere i risultati migliori. Varie alternative sono state vagliate nel corso della realizzazione del progetto ma, data la natura particolarmente ostica dei dati, non sempre sono risultati modelli con performance sufficienti. Il progetto si articola in 3 cartelle:

- results: dove troviamo le immagini delle curve di apprendimento dei modelli, oltre che i file dei migliori modelli ottenuti. In questo caso si è deciso di salvare soltanto una tipologia di modello (cnn) in quanto è quello che impiega più tempo per essere allenato;
- scripts: troviamo i file che contengono gli script per lanciare le varie fasi dell'apprendimento automatico;
- src: è la cartella che contiene i file con il codice sorgente, dalla creazione dei dataset alla definizione dei modelli. Inoltre troviamo anche una cartella utils che contiene: un file nel quale sono definite alcune costanti utilizzate in varie parti del codice, un file contenente alcune funzioni di debug usate durante lo sviluppo del progetto e un file .json contenente le feature che la libreria TSFEL va ad estrarre; infatti di default estrae tutta una serie di feature che non sono state ritenute necessarie.

4.1.1 Strumenti Utilizzati

Per realizzare i modelli sono state utilizzate due librerie:

- scikit-learn;
- tensorflow.

In particolare scikit-learn è stata utile anche nella fase di valutazione, in quanto implementa strumenti che consentono di calcolare varie metriche. Mentre per la realizzazione dei grafici è stata utilizzata la libreria matplotlib. Per la lettura e l'elaborazione dei segnali EEG è stata utilizzata la libreria MNE, oltre che la già citata TSFEL per il calcolo delle feature.

4.1.2 Valutazione

In questo caso si è deciso di considerare le seguenti metriche per la valutazione dei modelli:

- Accuratezza;
- Precisione;
- Richiamo;
- F1 score;
- Log Loss.

4.2 Apprendimento Supervisionato

Con il dataset ottenuto dal secondo approccio i risultati migliori sono stati raggiunti da due modelli, entrambi realizzati con scikit learn:

- Albero di Decisione, impostando la profondità massima a 3;
- Regressione Logistica.

Sono stati tentati anche altri modelli come RandomForestClassifier senza però ottenere delle performance accettabili. Di seguito verranno riportati solo i risultati dei primi due modelli.

4.2.1 Albero di Decisione

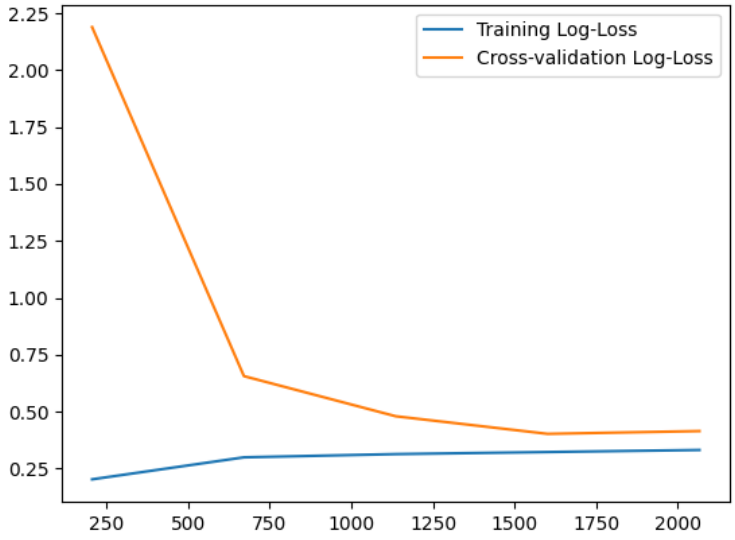
Con questo primo modello siamo riusciti a raggiungere le seguenti performance:

Accuracy	**Precision**	**Recall**	**F1 Score**	**Log Loss**
0.83	0.86	0.80	0.82	0.38

e la seguente matrice di confusione:

	actual positive	**actual negative**
predict positive	250	38
predict negative	59	229

Inoltre, per verificare che il modello appreso non sia il risultato di overfitting, è stata tracciata una curva di apprendimento attraverso la funzione `learning_curve` di scikit-learn, impostando una 10-fold cross validation. La curva della log-loss è la seguente:



Possiamo osservare come la differenza fra la log loss di addestramento e di cross-validation sia abbastanza ridotta, e che continua a diminuire con una quantità di dati più grande, stabilizzandosi dopo circa 1250 campioni. Quindi sembrerebbe che il modello non soffra eccessivamente di overfitting e che l'aggiunta di nuovi dati non migliorerebbe i risultati.

4.2.2 Regressione Logistica

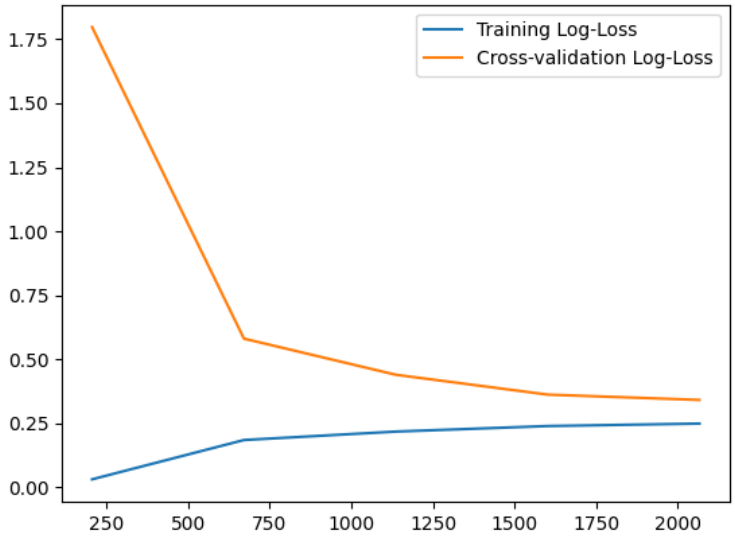
Per cercare di migliorare le performance del modello precedente, ho tentato un altro dei modelli "di base" descritti nel capitolo relativo all'Apprendimento Supervisionato del libro, ovvero la Regressione Logistica, ottenendo i seguenti risultati:

Accuracy	Precision	Recall	F1 Score	Log Loss
0.86	0.90	0.81	0.85	0.38

Possiamo notare un leggero miglioramento rispetto al modello precedente, ma nulla di significativo. Di seguito si riporta che la matrice di confusione che conferma i risultati di sopra:

	actual positive	actual negative
predict positive	262	26
predict negative	55	233

Di seguito possiamo osservare la curva di apprendimento, realizzata allo stesso modo del modello precedente:



dove anche in questo caso possiamo notare come la curva si stabilizzi dopo circa 1250 campioni, indicando che l'aggiunta di dati difficilmente migliorerebbe le prestazioni. Anche la log-loss finale risulta piuttosto bassa e vicina a quella di training, indicando che l'overfitting se c'è, è minimo.

4.2.3 Rete Neurale Convoluzionale

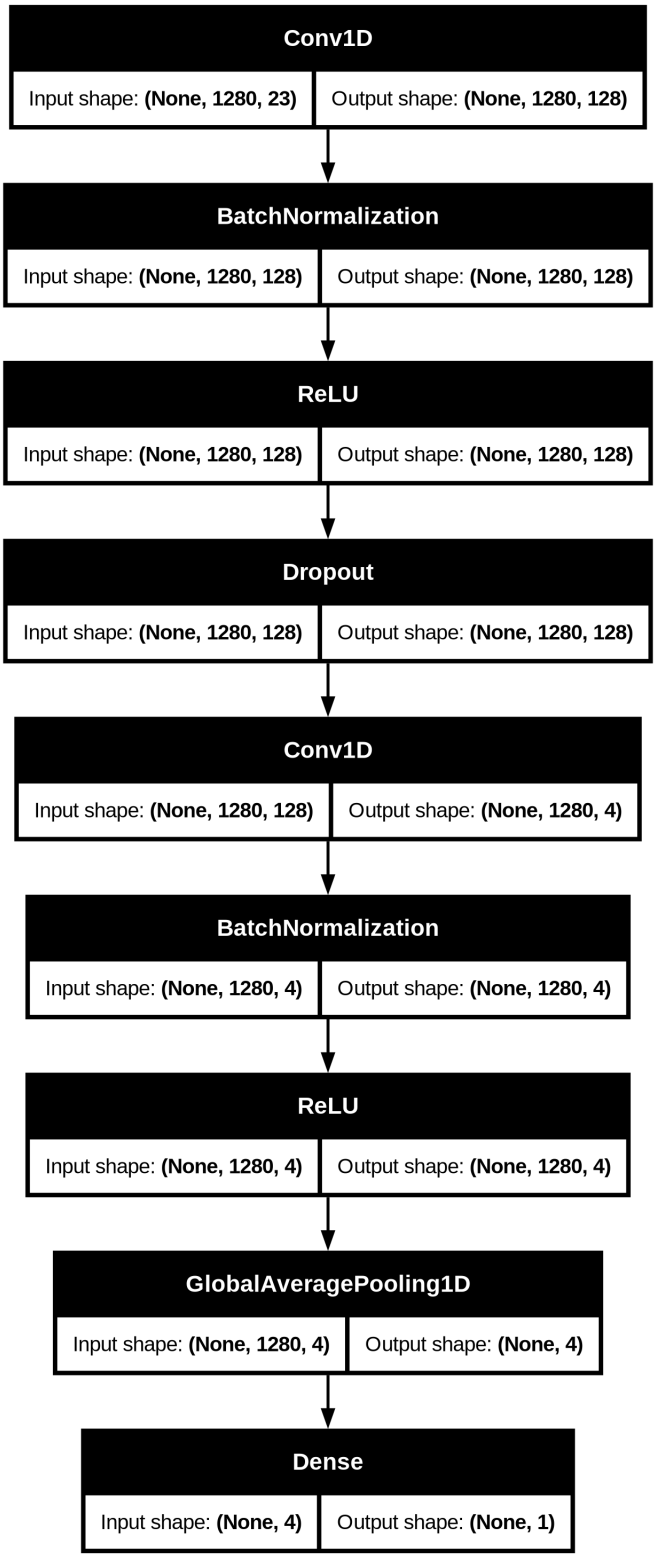
Per cercare di migliorare ulteriormente le prestazioni è stata realizzata una CNN allenata direttamente sulle serie temporali del dataset ottenuto nel primo approccio. L'idea è nata prendendo ispirazione dall'esempio 8.7 del capitolo 8.4 dedicato proprio alle CNN. L'esempio mostra come l'uso di un kernel monodimensionale possa semplificare notevolmente un segnale, rendendo più semplice individuare alcune caratteristiche che consentono di prevederne la classe di appartenenza.

Le CNN sono un tipo di Rete Neurale Feedforward molto usate nella computer vision, nell'elaborazione di immagini e anche nell'analisi di serie temporali. Elemento fondamentale di questo tipo di Reti neurali è il kernel, che è un array utilizzato per eseguire l'operazione di convoluzione sull'input. Solitamente sono array bidimensionali nei casi in cui il problema coinvolge, per esempio, delle immagini ma possono anche essere monodimensionali nei casi in cui si vanno ad analizzare delle serie temporali, come in questo caso. Sono detti "operatori lineari appresi" perché i loro valori vengono ottimizzati durante il processo di addestramento, utilizzando i dati per adattarsi a specifici pattern o caratteristiche presenti nel problema da risolvere. Il loro scopo è estrarre caratteristiche rilevanti dai dati in input.

A differenza di una Rete Neurale classica, quindi composta da unità dense, gode di due proprietà molto importanti che la rendono appetibile per compiti quali il riconoscimento di un oggetto in un immagine, e sono:

- località: i parametri che compongono i kernel sono appresi in modo da catturare relazioni locali tra i vicini, invece di essere basati su tutte le unità come accade in un layer completamente connesso.
- condivisione dei parametri: per un singolo kernel, gli stessi parametri sono usati in tutte le posizioni, questo permette alla rete di rilevare lo stesso pattern in qualsiasi punto della serie temporale.

Tutte queste caratteristiche sembrano adatte per la realizzazione di un modello che sia in grado di classificare correttamente un segmento di un segnale EEG. L'architettura del modello proposto possiamo vederla nell'immagine seguente:



Oltre ai layer convoluzionali, possiamo trovare anche dei layer che effettuano la normalizzazione dei batch, questo per normalizzare l'output del layer convoluzionale precedente e quindi consentire un analisi più efficiente dei dati. A seguito di un layer BatchNormalization troviamo una funzione di attivazione, in questo caso una ReLU. Prima del layer di output troviamo un Pooling Layer, in particolare un Global Average Pooling, che non ha parametri da apprendere ma è una funzione che viene applicata ai risultati delle unità precedenti, calcolando il valore medio per ciascuna feature.

Per prevenire il problema dell'overfitting, il modello è stato valutato utilizzando un validation set pari al 20% del training set. Il training è configurato per salvare i pesi corrispondenti alla loss più bassa registrata sul validation set durante l'addestramento. Sono previste fino a 500 epoche di

addestramento, ma è stato implementato un meccanismo di early stopping: se la loss sul validation set non migliora per 50 epoche consecutive, il training viene interrotto automaticamente. Un ulteriore aspetto cruciale riguarda la gestione del learning rate. Il tasso di apprendimento iniziale è impostato a 0.001, come di default su keras, e viene ridotto automaticamente ogni 20 epoche in cui non si osserva un miglioramento sulla loss del validation set. Il training set è inoltre suddiviso in batch da 32 elementi, per migliorare l'efficienza dell'addestramento e consentire una migliore convergenza del modello. Sono stati utilizzati gli iperparametri di default di keras, usando come ottimizzatore Adam. Adaptive moments o Adam, in base alla documentazione di keras è un ottimizzatore adatto per problemi che sono grandi in termini di parametri, come in questo caso.

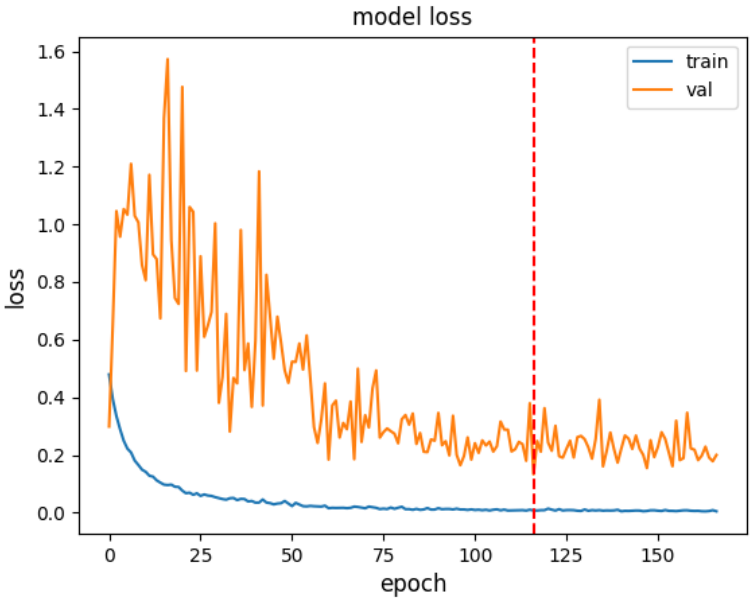
I risultati ottenuti sul test set sono i seguenti:

Accuracy	**Precision**	**Recall**	**F1 Score**	**Log Loss**
0.98	0.99	0.96	0.98	0.049

e la matrice di confusione è la seguente

	actual positive	**actual negative**
predict positive	287	1
predict negative	10	278

Quindi un miglioramento di tutte le metriche piuttosto netto rispetto ai due modelli precedenti. Per quanto riguarda la curva di apprendimento possiamo notare, soprattutto nelle prime epoche, un evidente oscillazione che tende a diminuire e a stabilizzarsi con il tempo. Vari tentativi sono stati effettuati per diminuire questa oscillazione e questo è il risultato migliore che si sia riusciti ad ottenere. La linea rossa indica l'epoca in cui il modello è stato salvato, in quanto ha raggiunto la log-loss più bassa registrata. Dopo questa linea possiamo notare come il train loss resti molto basso a differenza del validation loss che sembrerebbe non diminuire ulteriormente, suggerendo un possibile presenza di overfitting.



4.3 Conclusioni

Sicuramente dei tre modelli realizzati l'ultimo risulta quello con le performance migliori, ma la curva di apprendimento non convince del tutto e sicuramente ulteriori miglioramenti possono essere appli-

cati, sia al training set con un preprocessing più spinto per cercare di pulire i dati da eventuali segnali di disturbo, sia alla tipologia di modello, magari usandone uno più adatto alle serie come RNN.

Bibliografia

[1] World Health Organization. Epilepsy. https://www.who.int/news-room/fact-sheets/detail/epilepsy, 2024.

[2] Hisham Daoud and Magdy A. Bayoumi. Efficient epileptic seizure prediction based on deep learning. *IEEE Transactions on Biomedical Circuits and Systems*, 13(5):804–813, 2019.

[3] John Guttag. CHB-MIT Scalp EEG Database (version 1.0.0), 2010.

[4] Anupreet Singh and Sridhar Krishnan. Trends in eeg signal feature extraction applications. *Frontiers in Artificial Intelligence*, 5:1072801, 01 2023.