UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Laurea Triennale in Ingegneria Informatica

eXplainable Artificial Intelligence: Sviluppo di una web app per la ricostruzione del battito cardiaco partendo dall'EEG tramite una Rete Neurale Convoluzionale

12 Dicembre 2024

Relatore:

Prof: Antonio Luca Alfeo

Prof: Mario G.C.A. Cimino

Candidato:

Lorenzo Monaci

ANNO ACCADEMICO 2023/2024

Indice

Abstract

In medicina sorge spesso la necessità di sottoporre i pazienti a diversi esami per riuscire a stabilire una diagnosi, i quali possono essere dispendiosi in termini economici, di energia e tempo. Quindi ci chiediamo: *è possibile estrarre più informazioni da un singolo esame?* e qualora lo fosse, *come si può svolgere l'interazione con l'utente finale?*

Il nostro progetto di tesi ha l'obiettivo di rispondere (positivamente) a entrambe le domande. Si propone dunque un metodo per l'estrazione del segnale del battito cardiaco HR (Heart Rate) a partire dall'analisi dell' Elettroencefalogramma (EEG) utilizzando una Rete Neurale Convoluzionale (CNN), e lo sviluppo di un'applicazione che possa rendere fruibile questa tecnica agli utenti finali.

Si è scelto di utilizzare questo tipo di rete neurale perché diversi studi ne hanno dimostrato l'efficacia nell'analisi di dati correlati temporalmente. Per quanto riguarda lo sviluppo del'applicazione si fa uso di Google Colab, un servizio di Notebook Jupyter che offre libero accesso a risorse di calcolo assieme a Streamlit, un framework per lo sviluppo di applicazioni web interattive che fa uso del linguaggio di programmazione Python e infine Ngrok, un servizio di tunneling gratuito che ci permette di creare un link pubblico a cui gli utenti possono collegarsi.

La rete neurale è in grado di ricostruire il battito cardiaco dei singoli soggetti con elevata precisione e di soggetti multipli con una modesta accuratezza partendo da EEG filtrati dall'interferenza di segnali fisiologici maggiori (come il movimento degli arti) ma comunque affetti da rumore generato da piccoli movimenti (come il movimento involontario degli occhi o la respirazione). L'applicazione si interfaccia con l'utente in modo da ricevere come input dei parametri per l'analisi e file in formato csv (comma separated values) gli EEG di un numero variabile di pazienti, così da addestrare la rete in modo da generare un modello che sia il più possibile generico. Si propone inoltre all'utente i risultati dell'elaborazione della CNN e le prestazioni di quest'ultima, in forma di grafici

Capitolo 1

Introduzione

Il nostro progetto punta a sviluppare un'applicazione web per ottenere il battito cardiaco dei pazienti di cui analizziamo l'elettroencefalogramma utilizzando una **Rete Neurale Convoluzionale**.

1.1 Reti Neurali Convoluzionali, un approccio vantaggioso

Questa tecnica è molto efficace per via delle elevate prestazioni di questo tipo di reti per l'analisi di segnali temporali. Un software del genere permetterebbe di accorciare notevolmente le tempistiche e i costi delle visite mediche nonché fornire informazioni che normalmente vengono estratte da esami separati, si pensi quindi all'estensione di questa tecnica per la ricostruzione di altri segnali fisiologici, come la pressione sanguigna o la fotopletismografia [1].

1.2 Impieghi nella realtà, l'XAI a supporto della scienza

É ovvio pensare che un oggetto come questo susciterebbe grande interesse all'interno di vari campi della medicina *(ad es. cardiologia, neurologia, . . .)*, ma si può pensare anche a soggetti interessati allo studio di *Machine Learning* e in particolare all'implementazione di quest'ultimo in ambito medico.

Prendiamo come esempio per il primo caso i medici di base, il loro lavoro sarebbe sicuramente semplificato da questo strumento in quanto potrebbero concentrarsi su un singolo esame a cui sottoporre vari pazienti e ricevere come risposta a questi sia HR che EEG.

Invece immaginiamo degli studenti di machine learning, potrebbero prendere spunto per studi futuri o basare ricerche sul nostro caso di studio.

[1] Un fotopletismogramma (FPG) è un pletismogramma ottenuto per via ottica che può essere usato per rilevare variazioni del volume sanguigno all'interno di tessuti microvascolari. [Fonte:Wikipedia]

Capitolo 2

Related Works

L'utilizzo delle CNN è fondamentale per una predizione accurata nel caso di serie temporali, dato che queste sono 'comprese' in modo molto preciso grazie appunto alla convoluzione che viene applicata, che nel caso di segnali correlati temporalmente permette di fornire una descrizione molto precisa di questi ultimi.

Dato che l'EEG è appunto un segnale temporale, si sceglie di analizzarlo utilizzando queste reti per, nel nostro caso estrarre il battito cardiaco, ma come già detto in precedenza potremmo concentrarci su qualsiasi altro segnale fisiologico.

2.1 L'analisi delle serie temporali

Come dimostrato in uno studio pubblicato sul Journal of Big Data: 'Time-series analysis with smoothed Convolutional Neural Network' *(Aji Prasetya Wibawa, Agung Bella Putra Utama)*[1] l'impiego delle CNN non è sempre pensato per analizzare serie temporali, infatti nasce come approccio all'analisi delle immagini.

Infatti le CNN necessitano di dati filtrati da rumori troppo evidenti per avere delle predizioni consistenti, quindi nasce la necessità di rendere i dati *smooth*, cioè adatti a essere dati in input alla rete.

2.2 Applicazione a segnali fisiologici

I segnali elettrici che il cervello emette e che vengono quindi campionati dai vari sensori dell'apparecchio sono spesso misti a vari tipi di rumore, da quello più grande in ampiezza e facile da riconoscere e quindi rimuovere *(movimenti articolari, rumore termico, ...)* a quello lieve di movimenti impercettibili e involontari del nostro corpo *(Movimento delle palpebre, respiro, organi interni, ...)*.

La maggior parte di queste interferenze possono essere facilmente eliminabili tramite tecniche di *filtraggio*, ma come analizzato nello studio pubblicato sul Journal of Computer Science and Information Technology: 'Convolutional Neural Network Application in Biomedical Signals' *(Haya, A.)*[2] ci sono alcuni artefatti che non possono essere rimossi (rumore termico) o individuati facilmente (micro-movimenti), quindi è necessario

far si che la rete neurale 'si abitui' alla presenza di questi disturbi e li etichetti come tali.

2.3 Il nostro approccio

Cerchiamo di combinare le due cose e rendere in primis i dati privi di ampi artefatti di rumore grazie a operazioni di *filtraggio in frequenza*, successivamente si applica una divisione tra dati di testing e training, utilizzando *un solo dataset* per la validazione e il resto come dataset di addestramento. Ciò permette di dotare la rete di una visione più 'ampia' del problema e generare così un modello che riesca a adattarsi a qualunque soggetto. In seguito si applica una *normalizzazione* ai dati, facendo si che abbiano *media* nulla e *varianza* unitaria in modo da ridurre la variabilità dei dati.

2.3.1 Rappresentazione dell'output

'Keep it simple, keep it stupid'[3]

In base a questo principio alla base di molte applicazioni informatiche e ingegneristiche scegliamo una tecnica di rappresentazione dell'output basata sui colori, in modo che possa essere colta al volo la differenza tra una predizione giusta e una sbagliata e dove si trova il picco cardiaco se presente.

Capitolo 3

Design e Implementazione

3.1 Costruzione del modello di rete neurale

Si tratta di un modello sequenziale composto da 5 *layer*:

→ **Conv1D**: un layer convoluzionale caratterizzato da 64 neuroni e una finestra di 5 campioni per volta impegnati nella convoluzione

→ **MaxPooling1D**: un layer che riduce il numero di campioni prendendo quello con valore massimo in un pool di valore specificato, nel nostro caso 2

→ **Flatten**: layer che si occupa di appiattire la dimensione del tensore in esame, rendendolo monodimensionale

→ **Dense**: due layer con filtri densamente connessi[1] con funzioni di attivazione *rettificatrice* e *sigmoidea* riepsttivamente con un numero di filtri pari a 128 e 1.

Il modello è poi compilato con un `batch size` pari a 64 e impostato per andare avanti fino a 100 epoche.

```
1     from tensorflow.keras.models import Sequential
2     from tensorflow.keras.layers import Conv1D, MaxPooling1D
      , Flatten, Dense
3
4     # ...
5
6     model = Sequential()
7     model.add(Conv1D(filters=64, kernel_size=5, activation='
      relu', input_shape=(n_points, n_series)))
8     model.add(MaxPooling1D(pool_size=2))
9     model.add(Flatten())
10    model.add(Dense(128, activation='relu'))
11    model.add(Dense(1, activation='sigmoid'))
12
13    model.compile(loss='binary_crossentropy', optimizer='
      adam', metrics=['accuracy', f1_m, precision_m, recall_m
      ])
14
15    # ...
```

[1] Ogni filtro è connesso a tutti qulli del layer precedente e successivo

Questo design è efficiente per il nostro caso di studio in quanto rimane un modello relativamente semplice e con un numero limitato di epoche per evitare l'overfitting (stiamo analizzando dataset relativamente piccoli, neanche vicini ai **Big Data**[2]), e un batch di dimensioni intermedie per analizzare una modesta quantità di dati per volta[3].

3.2 Data Preprocessing

I dati inseriti dall'utente in formato `csv` vengono trattati in due modi differenti a seconda del numero di soggetti che stiamo analizzando, per garantire l'analisi di un particolare soggetto oppure per cercare di generare un modello generico.

3.2.1 Soggetto singolo

In questo caso si adotta uno split del dataset stesso seguendo il Principio di Pareto[4]. Così facendo addestriamo la rete neurale su un soggetto particolare, quindi il modello sarà creato 'ad-hoc' per quello specifico soggetto.

3.2.2 Soggetti multipli

Analizzando più soggetti alla volta consideriamo il fatto di addestrare la rete su tutti i soggetti meno uno per ogni soggetto, utilizzando quest'ultimo come soggetto di test.

Successivamente alla fase di split, ogni dataset viene normalizzato tramite la funzione `transform()` di `StandardScaler` del paccetto `sklearn.preprocessing`.

```
1  import nuympy as np
2
3  # ...
4
5  for i in range(n_soggetti):
6      if n_soggetti < 2:
7          LOG("Loading data and labels")
8
9          multiple_subj = False
10
11         X = data[0].values.reshape(n_samples, n_points,
       n_series)
12         y = labels[0].values
13
14         test_idx_end = int((n_samples // 2) + int(n_samples
       * 0.1))
15         test_idx_begin = int((n_samples // 2) - int(
       n_samples * 0.1))
16
17         # Create a boolean mask to select elements to keep
       for training and testing sets
```

[2] I quali si considerano tali per grandezze del centinaio di TB

[3] Si ricorda la regola di Pareto[4], per la quale l'80% dei risultati proviene dal 20% dei dati

```
18          mask_train = np.ones(X.shape[0], dtype=bool)
19          mask_train[test_idx_begin:test_idx_end] = False
20          X_train = X[mask_train, :, :]
21          y_train = y[mask_train]
22
23          mask_test = np.zeros(X.shape[0], dtype=bool)
24          mask_test[test_idx_begin:test_idx_end] = True
25          X_test = X[mask_test, :, :]
26          y_test = y[mask_test]
27
28          X_train_2D = X_train.reshape(X_train.shape[0], -1)
29          X_test_2D = X_test.reshape(X_test.shape[0], -1)
30
31          # Apply StandardScaler
32          X_train = scaler.fit_transform(X_train_2D)
33          X_test = scaler.transform(X_test_2D)
34
35          # Reshape back to 3D for CNN
36          X_train = X_train.reshape(X_train.shape[0], n_points
    , n_series)
37          X_test = X_test.reshape(X_test.shape[0], n_points,
    n_series)
38
39          LOG("Data and labels loaded!")
40          LOG(f'Train data: {X_train.shape}, test data: {
    X_test.shape}, train labels: {y_train.shape} test labels
    : {y_test.shape}')
41      else:
42          #---------------------------DATA GENERATION
    -------------------------------#
43          if n_soggetti > 1:
44              n_samples_overall = 0
45              X_train = []
46              X_test = []
47              LOG(f"Loading data of subject {i+1}...")
48              for j in range(n_soggetti):
49                  # Dataset di test
50                  if j == i:
51                      X_test = data[i].values
52                      X_test = scaler.fit_transform(X_test)
53                      X_test = X_test.reshape(n_samples,
    n_points, n_series)
54                      continue
55
56                  # Dataset di training
57                  X_train.extend(data[j].values)  # append to
    the whole time series
58                  n_samples_overall += int(data_files[j].name
    [-11:-7])
59
60              X_train = scaler.fit_transform(X_train)
61              X_train = X_train.reshape(n_samples_overall,
    n_points, n_series)
62
63          LOG("Data loaded!")
64          LOG(f'Train data: {X_train.shape}')
65          LOG(f'Test data: {X_test.shape}')
66
67          #---------------------------LABEL GENERATION
    -------------------------------#
68
```

```
69          y_train = []
70          y_test = []
71          first_iter = True
72          LOG(f"Loading labels of subject {i+1}...")
73
74          for j in range(n_soggetti):
75              # Label di testing
76              if j == i:
77                  y_test = labels[i].values
78                  continue
79              # Label di training
80              if first_iter:
81                  y_train = labels[j].values
82                  first_iter = False
83              else:
84                  y_train = np.append(y_train, labels[j].
    values, axis=0)
85
86          LOG("Labels loaded!")
87          LOG(f'Training labels: {y_train.shape}')
88          LOG(f'Testing labels: {y_test.shape}')
89  # ...
```

Appendice A

Questa è un'appendice

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Appendice B

Questa è un'altra appendice

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Bibliografia

[1] Agung Bella Putra Utama Aji Prasetya Wibawa. Time-series analysis with smoothed convolutional neural network. *Journal of Big Data*, 1:18, 04 2022.

[2] A. Haya. Convolutional neural network application in biomedical signals. *ResearchGate*, 1:16, 01 2018.

[3] U.S. Navy, 1960.

[4] Vilfredo Pareto, 1848-1923.