Università degli studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (classe L-31)

Convolutional neural networks for seizure prediction using intracranical and scalp electroencephalogram

Progetto di Gruppo

Tutor:
Prof.Marco Piangerelli

Gruppo:
Simone Morettini
Alessandra Renieri

ANNO ACCADEMICO 2017-2018

Indice

Introduzione

Il progetto "Studio sulle CNNs per la predizione di crisi epilettiche", descritto in questa relazione, è stato portato avanti da Simone Morettini e Alessandra Renieri, sotto la supervisione del Dott.Marco Piangerelli. L'obiettivo è quello di emulare i risultati proposti nell'articolo *Convolutional neural networks for seizure prediction using intracranical and scalp electroencephalogram* [1], per poi creare un codice open source di data analisi, pubblicato in Github (https://github.com/MesSem/CNNs-on-CHB-MIT).

Durante lo svolgimento del progetto è stato utilizzato il linguaggio Python, ed in particolare le librerie Keras (e Tensorflow).

Il progetto è stato suddiviso in tre macro workpackages:

- analisi dataset:
  - studio del dataset;
  - spettrografia del dataset;
  - modulazione degli spettri ottenuti;
- CNNs (studio teorico e applicazione pratica):
  - studio teorico;
  - bilanciamento del dataset;
  - creazione della rete;
  - allenamento della rete;
  - produzione dei risultati;
- risultati:
  - analisi dei risultati;
  - comparazione con i risultati ottenuti nell'articolo.

Il loro svolgimento risulta essere stato in linea con la programmazione temporale definita all'inizio del lavoro.

Capitolo 1

Stato dell'arte

1.1 Epilessia

L'epilessia è una malattia neurologica che colpisce circa 50 milioni di persone in tutto il mondo. In alcuni casi viene definita cronica e in altri transitoria (i.e. un episodio mai più ripetutosi), ed è caratterizzata da ricorrenti e improvvise manifestazioni con improvvisa perdita di coscienza e violenti movimenti convulsi dei muscoli, dette *crisi epilettiche*. In base alla zona del cervello coinvolta nella crisi, i pazienti possono fare esperienza di sintomi differenti: crisi di durata molto breve (tanto da passare inosservate), o lunghi periodi. Le crisi sono definite come improvvisi picchi di attività elettrica nel cervello. Secondo l'OMS una crisi epilettica non è sinonimo di epilessia: il 10% della popolazione mondiale ha un attacco epilettico durante l'arco della sua vita.

Nel 2005 l'epilessia era stata definita, dal punto di vista concettuale, come un disturbo cerebrale caratterizzato da una persistente predisposizione a sviluppare crisi epilettiche. Nella pratica clinica questa definizione viene solitamente applicata quando si manifestano due crisi epilettiche non provocate, separate da un intervallo di tempo maggiore di 24 ore. La International League Against Epilepsy (ILAE) ha ora accettato le raccomandazioni di una Task Force che prevedono la modifica della definizione pratica in circostanze particolari che non soddisfano i criteri di due crisi epilettiche non provocate. La Task Force ha proposto che l'epilessia debba essere considerata una malattia cerebrale definita da una delle seguenti condizioni:

- almeno due crisi non provocate (o riflesse) verificatesi a meno di 24 ore di distanza;

- una crisi non provocata (o riflessa) e una probabilità di ulteriori crisi simile al rischio generale di recidiva (almeno 60%) dopo due crisi non provocate, nei successivi 10 anni;
- diagnosi di una sindrome epilettica.

Si considera risolta l'epilessia nei soggetti che hanno avuto una sindrome epilettica età-dipendente ma che al momento attuale hanno superato questo limite di età o nei soggetti che sono rimasti liberi da crisi negli ultimi 10 anni in assenza di farmaci antiepilettici per almeno gli ultimi 5 anni. Il concetto di "risolto" non è necessariamente sinonimo di quello convenzionale di "remissione" o "guarigione". Definizioni pratiche diverse possono essere formulate e utilizzate per svariati scopi specifici. Questa definizione rivisitata di epilessia allinea il termine con l'utilizzo pratico. Nel 2015 l'ILAE ha classificato tre tipi di attacchi epilettici:

- *convulsioni focalizzate o parziali*: le crisi hanno la loro origine da una popolazione neurale confinata in una parte di un emisfero cerebrale;
- *convulsioni generalizzate*: le crisi hanno la loro origine da una popolazione neurale che appartiene ad entrambi le parti dell'emisfero cerebrale;
- *crisi non classificate*: non esistono argomenti per classificare le crisi.

Ogni categoria mostra alcune caratteristiche le la distingue dalle altre.

Il trattamento di un paziente con crisi epilettiche dipende dal tipo specifico di crisi e dalla sindrome epilettica. Non esiste cura per l'epilessia, ma circa il 70% dei pazienti trova beneficio usando i farmaci antiepilettici (AED): farmaci che hanno lo scopo di sedare le convulsioni. Esistono casi in cui nemmeno gli AED possono controllare le convulsioni.

Alternativa ai farmaci è la chirurgia, ovvero la rimozione della zona epilettogena. Tale zona non è chiaramente identificata, deve essere piccola e non deve interferire con le altre funzioni cerebrali. Per questo motivo non è possibile procedere sempre con la chirurgia.

Negli ultimi anni i miglioramenti tecnologici hanno consentito la diffusione di dispositivi impiantabili che si possono utilizzare sia per la diagnosi e sia per meccanismi di soppressione della crisi (come ad essere un pacemaker cerebrale).

La tecnica standard per la registrazione dell'attività elettrica cerebrale è l'EEG: si posizionano sul cuoio capelluto più elettrodi i quali misurano l variazioni di tensioni dovute al flusso di correnti ioniche all'interno del cervello. Un dispositivo più invasiva è l'iEEG o ECoG intracranico: le registrazioni

vengono fatte usando una griglia di elettrodi che non sono posti sulla superficie della corteccia, ma sono inseriti nel cervello (che penetrano di una certa profondità nel cervello).

1.2 Convolutional neural network (CNN)

Prima di addentrarci nella struttura delle Convolutional Neural Networks è necessario richiamare la definizione di una rete neurale classica - modelli ad elevato numero di parametri ispirati all'architettura del cervello umano e promossi come approssimatori universali (sistemi che, se alimentati da una sufficiente quantità di dati, sono in grado di apprendere qualsiasi relazione predittiva. Sono state inventate intorno alla metà degli anni 80, hanno goduto di un periodo di popolarità relativamente breve, per poi essere riprese attorno al 2010, grazie al notevole miglioramento delle risorse computazionali, per problemi come classificazioni di immagini e video e il riconoscimento testuale/vocale.

Il neurone è l'elemento basilare del sistema nervoso, ed anche il più importante, tanto che può essere considerato l'unità di calcolo primaria alla base della nostra intelligenza. Il sistema nervoso umano ne è costituito approssimativamente da 86 miliardi, connessi fra di loro da un numero di sinapsi dell'ordine di 10^{15}. Ogni neurone riceve in input il segnale dai suoi dendriti e, una volta elaborato, produce un segnale di output lungo il suo unico assone, che una volta diramatosi lo collega ai neuroni successivi attraverso le sinapsi. Questa attività biologica può essere rappresentata da un modello matematico: l'idea è che l'insieme delle sinapsi possa essere in qualche modo appreso e sia in grado di controllare, in base al segno e all'intensità, l'influenza di un neurone sugli altri.

Nel modello base i dendriti portano il segnale al corpo della cellula, dove sono sommati: se il risultato di questa somma supera una certa soglia, il neurone invierà un impulso attraverso il suo assone. La tempistica degli impulsi non viene considerata rilevante, e si assume pertanto che l'informazione sia portata solamente dalla somma di quest'ultimi. Tale somma viene modellata da quella che viene chiamata funzione di attivazione, che tipicamente coincide con la funzione sigmoide $\sigma(x) = \frac{1}{1+e^{-x}}$.

In altre parole, ogni neurone esegue un prodotto vettoriale tra i suoi input e il suo set di pesi, somma un termine di distorsione e applica una funzione di attivazione non lineare (in caso contrario la rete neurale si ridurrebbe ad un modello lineare generalizzato).

Nella figura vediamo rappresentato il diagramma di una semplice rete neurale con 4 predittori (x_j), un singolo strato nascosto (hidden layer L_2),

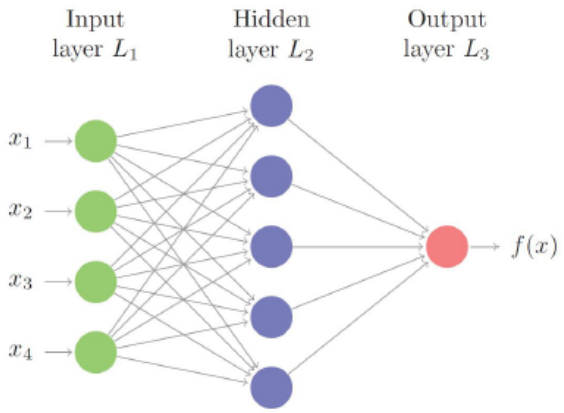


Figura 1.1: Diagramma di una rete neurale con un singolo strato nascosto

composto da 5 neuroni ($a_i = g(w_{i_0}^{(1)} + \sum_{j=1}^{4} w_{l_2}^{(1)} x_j)$) e una singola unità in uscita (output layer) ($y = h(w_0^{(2)} + \sum_{i=1}^{5} w_i^{(2)} a_i)$). Ogni neurone a_i è connesso allo strato di input attraverso il vettore di parametri/pesi (w). L'idea è quella che ogni neurone apprenda una funzione semlice binaria on/off (il lancio dell'impulso), e proprio per questo che chiamo g *funzione di attivazione*.

Le Convolutional Neural Networks (CNNs) sono reti neurali specializzate nel processamento di dati. Il nome Convolutional Neural Networks deriva dal fatto che tali reti utilizzano un'operazione matematica lineare chiamata convoluzione. Di conseguenza, una rete neurale classica che implementa operazioni di convoluzione in almeno uno dei suoi strati, viene definita Convolutional. Gli strati composti da operazioni di convoluzione prendono il nome di Convolutional Layers, ma non sono gli unici strati che compongono una CNN: la tipica architettura prevede infatti l'alternarsi di Convolutional Layers, Pooling Layers e Fully Connected Layers.

Quindi, le reti neurali tradizionali ricevono in input un singolo vettore, e lo trasformano attraverso una serie di strati nascosti, dove ogni neurone è connesso ad ogni singolo neurone sia dello strato precedente che di quello successivo (ovvero "fully-connected ") e funziona quindi in maniera completamente indipendente, dal momento che non vi è alcuna condivisione delle connessioni con i nodi circostanti. Nel caso l'input sia costituito da immagini di dimensioni ridotte, ad esempio $32 \times 32 \times 3$ (32 altezza, 32 larghezza, 3 canali colore), un singolo neurone connesso in questa maniera comporterebbe un numero totale di $32 \times 32 \times 3 = 3072$ pesi, una quantità abbastanza grande ma ancora trattabile. Le cose si complicano però quando le dimensioni si fanno importanti: salire ad appena 256 pixel per lato comporterebbe un

carico di $256 \times 256 \times 3 = 196608$ pesi per singolo neurone, ovvero quasi 2 milioni di parametri per una semplice rete con un singolo strato nascosto da dieci neuroni. L'architettura fully-connected risulta perciò troppo esosa in questo contesto, comportando una quantità enorme di parametri che condurrebbe velocemente a casi di sovradattamento. Inoltre, considerazione ancor più limitante, un'architettura di questo tipo fatica a cogliere la struttura di correlazione tipica dei dati a griglia. Le Convolutional Neural Networks prendono invece vantaggio dall'assunzione che gli input hanno proprio una struttura di questo tipo, e questo permette loro la costruzione di un'architettura su misura attraverso la formalizzazione di tre fondamentali proprietà: l'interazione sparsa (sparse interaction), l'invarianza rispetto a traslazioni (invariant to translation), e la condivisione dei parametri (weight sharing). Il risultato è una rete più efficace e allo stesso tempo parsimoniosa in termini di parametri.

La mappa di attivazioni è formata da neuroni connessi localmente allo strato di input attraverso i parametri del filtro (kernel) che li ha generati. L'estensione spaziale di questa connettività, che coincide con la dimensione del filtro, costituisce un iperparametro della rete e viene chiamata anche campo recettivo del neurone, o receptive field . Questa rappresenta la prima delle tre proprietà fondamentali delle Convolutional Neural Networks, ossia la *sparse interaction.*

Generalmente un Convolutional Layer è formato da un set di filtri molto più numeroso tutti con la medesima estensione spaziale. Durante lo stage feed-forward ogni filtro viene fatto convolvere lungo la larghezza e l'altezza del volume di input, e pertanto vengono prodotte tante mappe di attivazioni bidimensionali, quanti filtri, le quali forniscono ognuna la risposta del relativo filtro in ogni posizione spaziale. La loro concatenazione lungo la terza dimensione produce l'output del Convolutional Layer. I neuroni appartenenti alla stessa mappa di attivazione condividono sempre lo stesso set di pesi, ossia quelli del filtro che li ha generati. Questo definisce la seconda proprietà fondamentale delle Convolutional Neural Networks, ossia il *weight sharing.*

La disposizione finale il numero di neuroni che compongono il volume di output del Convolutional layer sono controllati da tre iperparametri:

- profondità - corrisponde al numero dei filtri che compongono lo strato;
- stride - specifica il numero di pixel di cui si vuole traslare il filtro ad ogni spostamento. Quando lo stride è 1 significa che ci stiamo muovendo il filtro di un pixel alla volta, valori più alti corrispondono a movimenti dei filtri con salti maggiori, che generano output di dimensioni minori;

- zero-padding - è spesso utilizzato per far combaciare la dimensione dell'input con quella dell'output e va a misurare lo spessore di un bordo di zeri che viene aggiunto al volume di input.

Larghezza e altezza del volume di output (O) possono essere calcolate come funzione della relativa dimensione nel volume di input (I), del campo recettivo del neurone (F), dello stride (S) applicato allo spostamento dei filtri, e della quantità di zero-padding (P) utilizzata per i bordi:

$$O = \frac{(I - F - 2P)}{S} + 1$$

La profondità coincide con il numero dei filtri utilizzati all'interno dello strato. Pertanto, dato in input un volume di dimensioni $W_1 \times H_1 \times D_1$ il convolutional layer produrrà un volume di output $W_2 \times H_2 \times D_2$ con

$$W_2 = \frac{(W_1 - F - 2P)}{S} + 1$$

$$H_2 = \frac{(H_1 - F - 2P)}{S} + 1$$

$$D_2 = numerofiltri$$

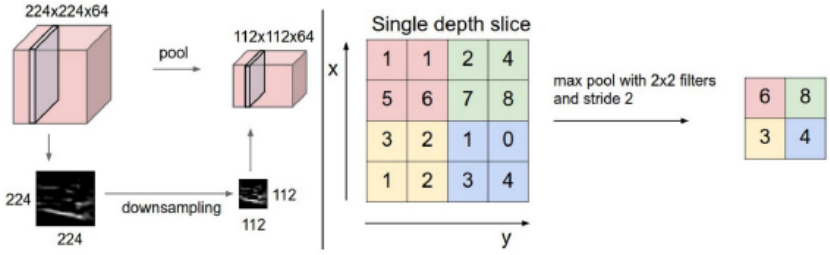
I valori di S e P devono essere scelti in modo che l'equazione di O restituisca un valore intero.

Nell'architettura di una CNN è pratica comune inserire fra due o più convolutional layers uno strato di Pooling, la cui funzione è quella di ridurre progressivamente la dimensione spaziale degli input (larghezza e altezza), in modo da diminuire numero di parametri e carico computazionale, e di conseguenza controllare anche il sovradattamento. Il Pooling Layer opera indipendentemente su ogni mappa di attivazioni applicando un filtro di dimensione $F \times F$ che esegue una determinata operazione deterministica (tipicamente il massimo o la media), e pertanto non comporta la presenza di pesi. Anche qui, il calcolo del volume finale dipende, oltre che dalle dimensioni $W_1 \times H_1 \times D_1$ di input, dai due iperparametri richiesti, ossia stride (S) ed estensione spaziale del filtro (F):

$$W_2 = \frac{(W_1 - F)}{S} + 1$$

$$H_2 = \frac{(H_1 - F)}{S} + 1$$

$$D_2 = D_1$$



Il Fully-Connected Layer è esattamente uguale ad un qualsiasi strato nascosto che compone le tradizionali reti neurali ed opera sul volume di output vettorizzato dello strato che lo precede. L'architettura fully-connected implica il rilassamento dell'assunzione di weight sharing : la funzione principale di tali strati, inseriti solo per ultimi a completare la struttura delle rete, è infatti quella di eseguire una sorta di raggruppamento delle informazioni ottenute negli strati precedenti, esprimendole attraverso un numero (l'attivazione neuronale) che servirà nei successivi calcoli per la classificazione finale. Intuitivamente, l'idea è quella che la rete apprenda filtri che si attivino alla visione di determinati tipi di caratteristiche (features) come ad esempio angoli, linee o blocchi di colore nello strato iniziale (features di basso livello), oppure combinazioni via via sempre più complesse negli strati superiori (features di alto livello).

Capitolo 2

Metodo proposto nell'articolo

La predizione delle crisi epilettiche è una delle applicazioni più interessanti per l'analisi dei dati predittivi, la cui utilità si potrebbe misurare in termini migliorativi della vita stessa dei pazienti con epilessia resistente ai farmici. Al fine di ottenere un'elevata sensibilità e un basso tasso di predizione falsa, molti studi si basano sull'estrazione di caratteristiche ah hoc per ogni paziente. Tale approccio non è generalizzabile e richiedere modifiche significative per ogni nuovo paziente (e ogni nuovo set di dati). L'approccio proposto dall'articolo utilizza l'applicazione delle reti neurali convoluzionali (CNNs), proponendo un metodo di predizione generalizzato. Si usa una trasformata di Fourier short time su finestre di EEG di 30 secondi, per estrarre informazioni sia nel dominio di frequenza (Hz) e del tempo (sec). L'algoritmo classifica i segmenti pre-ictali e interictali, tale metodo può essere applicato a qualsiasi paziente con qualsiasi set di dati. I dati usati sono: il dataset di EEG intracranici dell'Ospedale di Freiburg, il dataset di EEG del Boston Children's Hospital-MIT e il dataset di EEG intracranici dell'American Epilepsy Society Seizure Prediction Challenge. L'approccio proposto raggiunge una sensibilità dell'81,4%, 81,2% e 75% e un falso tasso di previsione di 0,06 / h, 0,16 / h, e 0,21 / h rispettivamente per ogni dataset.

2.1 Il dataset

Nel lavoro di ricerca descritto nell'articolo vengono usati tre datasets: il dataset di EEG intracranici dell'Ospedale di Freiburg, il dataset di EEG del Boston Children's Hospital-MIT (CHB-MIT) e il dataset di EEG intracranici dell'American Epilepsy Society Seizure Prediction Challenge.

Il dataset dell'Ospedale di Freiburg contiene dati da intracranical EEG (iEEG) di 21 pazienti, 13 usati realmente. Il tasso di campionatura è di 256 Hz; ci sono 6 canali di registrazione, tre dei quali inseriti nelle zone epilettogene e tre inserite in zone più lontane. Per ogni paziente ci sono almeno 50 minuti di registrazioni di dati pre-ictali e 24 ore di dati interictali.

Il CHB-MIT dataset contiene dati da scalp EEG (sEEG) di 23 pazienti pediatrici: 844 ore di registrazioni (quasi) continue e 163 crisi epilettiche. I segnali delle sEEG sono stati presi da 22 elettrodi ad un tasso di campionatura di 256 Hz. Si definisce [1] segnale interictale come quel periodo fra (almeno) 4 ore prima dell'inizio della crisi epilettica e 4 ore dopo la fine della crisi. Definisco, invece, segnale pre-ictale, quel segnale che registro dai 35 minuti ai 5 minuti prima della crisi epilettica. Inoltre occorre precisare che possono accadere crisi molto vicine le une dalle altre. In tal caso si considerano un'unica crisi, delle crisi che distano circa 30 min dalle crisi precedenti.

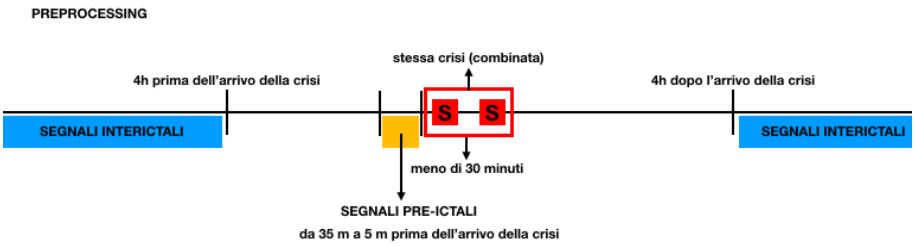


Figura 2.1: Segnali pre-ictali e interictali

Si sono ritenuti validi i pazienti che hanno meno di 10 crisi al giorno. In tal modo si hanno solo 13 pazienti per cui si hanno dati sufficienti. Il numero delle crisi considerate sono 64 e le ore interictali pari a circa 209.

Il dataset dell'American Epilepsy Society Seizure Prediction Challenge contiene dati da 5 cani e due pazienti con 48 crisi epilettiche e 627.7 ore di registrazioni di periodo interictale. Sui cani sono stati impiantati 16 elettrodi con tasso di campionatura a 400 Hz. Per i pazienti umani le campionature sono state differenti.

Dataset	EEG type	No. of patients	No. of channels	No. of seizures	Interictal hours
Freiburg Hospital	Intracranial	13 patients	6	59	311.4
Boston Children's Hospital–MIT	Scalp	13 patients	22	64	209
American Epilepsy Society Seizure Prediction Challenge (Kaggle)	Intracranial	5 dogs, 2 patients	16	48	627.7

Figura 2.2: I datasets

2.2 Prepocessing

Dal momento che viene usata una CNN, è necessario convertire i dati della EEG in una matrice (i.e. a formato immagine). La conversione deve mantenere intatte tutte le informazioni che arrivano dal segnale EEG. Nel lavoro si sono utilizzate le STFT (short-time Fourier transform), così da trasformare i segnali in una matrice a 2 dimensioni composta da un asse dei tempi e un asse delle frequenza. Viene utilizzata una finestra temporale di 30 s. Le registrazioni EEG sono contaminate da una powerlinenoise a 60 HZ (tipica nel continente americano). Quindi nel dominio delle frequenze è conveniente usare un filtro passa banda e togliere le frequenze nei ranges 57-63 Hz e 117-123 Hz, ovviamente viene rimossa che la componente a 0 Hz.

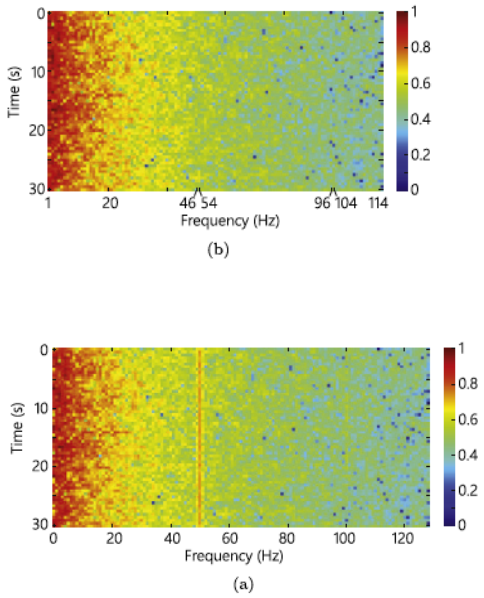


Figura 2.3: Esempi di STFT con finestra a trenta secondi prima (a) e dopo (b) la rimozione della linea a 60 Hz

Come possiamo vedere in Figura 1.1 risulta evidente che il dataset non è bilanciato: le registrazioni interictali sono molto di più rispetto a quelli preictali (e per ogni paziente abbiamo dei valori a sé). Per ovviare tale problema si andranno a creare dei dati sintetici mediante una tecnica particolare durante il momento dell'allenamento della rete. In pratica si andranno a creare dei dati "sintetici" facendo scorrere la finestra a 30-secondi lungo l'asse temporale di un fattore **S** nelle registrazioni dei segnali pre-ictali. **S** è scelto secondo il soggetto, in base al rapporto fra le varie ore pre-ictali e interictali. In tal modo si avranno

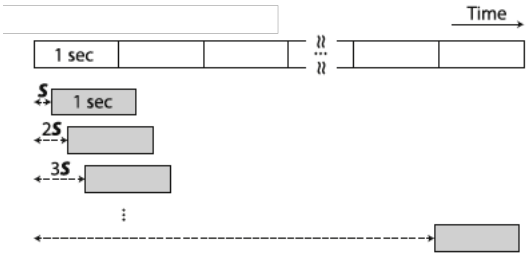


Figura 2.4: Generazione di dati pre-ictali sintetici

2.3 CNN

Il modello proposto è stato implementato usando Pyton 2.7 e Keras 2.0 (con la libreria Tensorflow). Nel lavoro è stata utilizzata una CNN a tre blocchi convoluzionali, ognuno dei quali consiste in una batch normalization (che standardizza gli input con media nulla e varianza unitaria - curva gaussiana), un layer convoluzionale con una ReLU e un layer maxpooling. Il primo blocco convoluzionale ha 16 kernel (filtri) di dimensione $n \times 5 \times 5$, con n numero dei canali EEG, con scorrimento (stride) di $1 \times 2 \times 2$. Il secondo blocco ha 32 kernel (filtri) di dimensione 3×3, con scorrimento (stride) di 1×1. Il terzo blocco ha 64 kernel (filtri) di dimensione 3×3, con scorrimento (stride) di 1×1. Il layer di maxpooling agisce su regioni 2×2

A seguire i tre blocchi convoluzionali ci sono due layers completamente connessi di grandezza di output 256 e 2 rispettivamente e aventi funzione di attivazione sigmoidale e softmax rispettivamente. Entrambi hanno un tasso di dropout dello $0,5$.

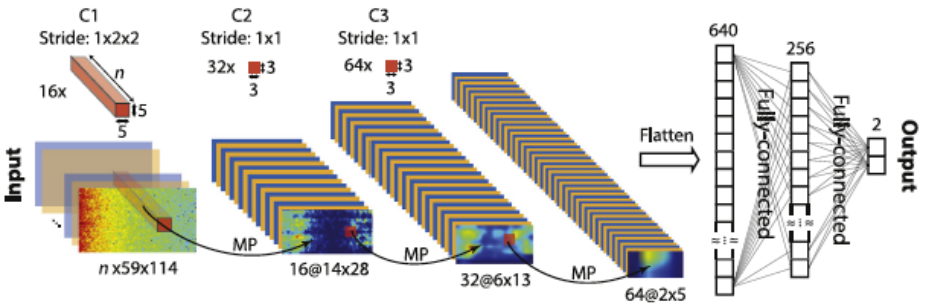


Figura 2.5: Architettura della CNN proposta nell'articolo

Al fine di non rendere il modello di predizione delle crisi epilettiche troppo specifico (problema di overfitting), si decide di utilizzare porzioni di segnale differenti nel training (training e validating) e nel testing: in particolare si selezionano rispettivamente il 75% e il 25%.

La valutazione del modello risulta molto robusto in quanto l'approccio di test del modello viene fatto secondo un modello *leave-one-out* cross-validation per ogni soggetto della ricerca. Se un paziente ha N segnali pre-ictali, allora $N-1$ di essi vengono usate per il training (training e validating) e il periodo che rimane viene usato per il testing. Questo viene fatto N volte, così da usare ogni segnale pre-ictale. Idem per i segnali interictali.

2.4 Postprocessing

Durante la fase di postprocessing occorre isolare i falsi positivi durante i periodi interictali. A tal fine si utilizza un filtro chiamato *k-di-n*: viene fatto suonare un allarme, che avverte di un arrivo di segnali pre-ictali, se e soltanto se almeno k predizioni su le n sono positive. Nella sperimentazione sono stati presi k= 8 e n= 10. Ciò significa che se durante gli ultimi 300 s almeno 240 s portano a una predizione positiva di arrivo di segnali pre-ictali, allora suona l'allarme.

2.5 System evaluation

Al fine dare una valutazione al sistema proposto, vanno definiti i valori di *sensitività* e *FPR* (false positive rate). Abbiamo allora bisogno di due oggetti:

- seizure occurrence period (SOP): è l'intervallo in cui ci si aspetta che accada la crisi; esso non deve essere troppo lungo, in modo da ridurre al minimo l'ansia nel paziente. Nel lavoro proposto è pari a 30 minuti.
- seizure prediction horizon (SPH): periodo fra l'allarme e l'inizio del SOP; esso deve essere un tempo lungo abbastanza per permettere interventi o precauzioni sufficienti per il paziente. Nel lavoro proposto è pari a 5 minuti.

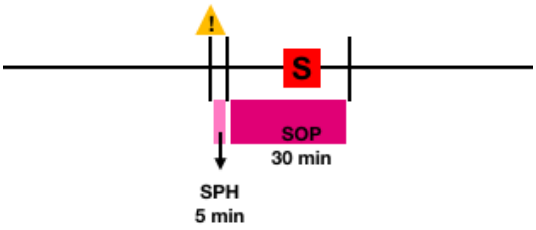


Figura 2.6: SOP e SHP

Una predizione è corretta se la crisi arriva dopo l'SPH ed entro l'SOP e un falso allarme c'è quando il sistema di predizione dà un risultato positivo ma nessuna crisi epilettica accade durante l'SOP.

La *sensitività* è definita come la percentuale di crisi epilettiche correttamente predette diviso il numero totale delle crisi, i.e. $\frac{TruePositive}{TruePositive+FalsePositive}$ L'FPR è definito come il numero di falsi allarmi per ora.

2.6 Risultati

Nella figura sottostante vengono descritti i risultati per i tre datasets:

Patient	No. of seizures	Interictal hours	Sensitivity (%)	FPR (/h)
Pat1	4	23.9	100 ± 0.0	0.00 ± 0.00
Pat3	5	23.9	100 ± 0.0	0.00 ± 0.00
Pat4	5	23.9	100 ± 0.0	0.00 ± 0.00
Pat5	5	23.9	40 ± 0.0	0.13 ± 0.00
Pat6	3	23.8	100 ± 0.0	0.00 ± 0.00
Pat14	4	22.6	50 ± 0.0	0.27 ± 0.00
Pat15	4	23.7	100 ± 0.0	0.02 ± 0.02
Pat16	5	23.9	80 ± 0.0	0.17 ± 0.13
Pat17	5	24	80 ± 0.0	0.00 ± 0.00
Pat18	5	24.8	100 ± 0.0	0.00 ± 0.00
Pat19	4	24.3	50 ± 0.0	0.16 ± 0.00
Pat20	5	24.8	60 ± 0.0	0.04 ± 0.00
Pat21	5	23.9	100 ± 0.0	0.00 ± 0.00
Total	59	311.4	81.4 ± 0.0	0.06 ± 0.00

Figura 2.7: Risultati del Freiburg Hospital - segnali iEEG

Patient	No. of seizures	Interictal hours	Sensitivity (%)	FPR (/h)
Pat1	7	17	85.7 ± 0.0	0.24 ± 0.00
Pat2	3	22.9	33.3 ± 0.0	0.00 ± 0.00
Pat3	6	21.9	100 ± 0.0	0.18 ± 0.00
Pat5	5	13	80 ± 20	0.19 ± 0.03
Pat9	4	12.3	50 ± 0.0	0.12 ± 0.12
Pat10	6	11.1	33.3 ± 0.0	0.00 ± 0.00
Pat13	5	14	80 ± 0.0	0.14 ± 0.00
Pat14	5	5	80 ± 0.0	0.40 ± 0.00
Pat18	6	23	100 ± 0.0	0.28 ± 0.02
Pat19	3	24.9	100 ± 0.0	0.00 ± 0.00
Pat20	5	20	100 ± 0.0	0.25 ± 0.05
Pat21	4	20.9	100 ± 0.0	0.23 ± 0.09
Pat23	5	3	100 ± 0.0	0.33 ± 0.00
Total	64	209	81.2 ± 1.5	0.16 ± 0.00

Figura 2.8: Risultati del CBH-MIT- segnali sEEG

Participant	No. of seizures	Interictal hours	Sensitivity (%)	FPR (/h)
Dog1	4	80	50 ± 0.0	0.19 ± 0.02
Dog2	7	83.3	100 ± 0.0	0.04 ± 0.03
Dog3	12	240	58.3 ± 0.0	0.14 ± 0.09
Dog4	14	134	78.6 ± 0.0	0.48 ± 0.07
Dog5	5	75	80 ± 0.0	0.08 ± 0.01
Pat1	3	8.3	100 ± 0.0	0.42 ± 0.06
Pat2	3	7	66.7 ± 0.0	0.86 ± 0.00
Total	48	627.7	75 ± 0.0	0.21 ± 0.04

Figura 2.9: Risultati del American Epilepsy Society Seizure Prediction Challenge- segnali iEEG

E' importante notare che nell'approccio del lavoro si riportano valori comparabili sia per i segnali iEEG che per quelli sEEG.

Capitolo 3

Il nostro risultato

L'obiettivo del nostro lavoro è quello di emulare i risultati proposti nell'articolo *Convolutional neural networks for seizure prediction using intracranical and scalp electroencephalogram* [1], per poi creare un codice open source di data analisi, pubblicato in Github (https://github.com/MesSem/CNNs-on-CHB-MIT).

Durante lo svolgimento del progetto è stato utilizzato il linguaggio Python 3.6.6, ed in particolare le librerie Keras 2.2.2 (e Tensorflow 1.10.0). Utilizziamo Anaconda per gestire tali pacchetti.

3.1 Il Dataset

In particolare, l'obiettivo era quello di emulare i risultati ottenuti utilizzando il CBH-MIT dataset. Studiando tale dataset abbiamo riscontrato diverse incoerenze nel calcolo delle ore interictali descritte nell'articolo e nella detenzione dei segnali pre-ictali o interictali. Abbiamo, quindi, deciso di concentrarci su quei pazienti per i quali tale problematiche erano ridotte (o quasi

assenti). In definitiva abbiamo scelto: Paziente 1, Paziente 2, Paziente 5, Paziente 19, Paziente 21, Paziente 23.

Precisiamo che per il Paziente 19 non abbiamo considerato la prima crisi epilettica e dunque non abbiamo considerato il periodo (molto breve) di segnale pre-ictale antecedente tale crisi. Inoltre abbiamo tolto tutto quel segnale che veniva registrato con meno di 22 canali.

3.2 Preprocessing

Il lavoro è avanzato in linea con quello proposto dall'articolo: dapprima abbiamo dovuto realizzare gli spettrogrammi dai dati reali che avevamo nel dataset. Il codice servito per tale fine è quello che si può leggere nel file "DataserToSpectogram.py". Nel codice viene utilizzata la libreria *scipy.signal*. La creazione degli spettrogrammi consiste in due fasi:

- "createSpec" - si crea lo spettrogramma canale per canale, singolarmente, dopo che la registrazione viene modulata. La modulazione viene fatta con le funzioni $butter_bandstop_filter$ e $butter_highpass_filter$ che tolgono le frequenze a 0 Hz e gli intervalli $57 - 63$ Hz e $117 - 123$ Hz.

- "createSpectogram" - divide il segnale in 30 secondi e poi crea lo spettrogramma per ogni canale, in quei 30 secondi. Tutti gli spettrogrammi vengono salvati in un unico vettore.

Sul disco viene salvato un file contenente una lista di 50 elementi (che rappresentano ognuno 30 secondi) in cui ciascun elemento contiene 22 spettrogrammi, uno per ogni canale.

L'output di tale pezzo di codice sono spettrogrammi pronti per essere dati in pasto alla rete.

```
def createSpec(data):
    fs=256
    lowcut=117
    highcut=123

    y=butter_bandstop_filter(data, lowcut, highcut, fs, order=6)
    lowcut=57
    highcut=63
    y=butter_bandstop_filter(y, lowcut, highcut, fs, order=6)

    cutoff=1
    y=butter_highpass_filter(y, cutoff, fs, order=6)

    Pxx=signal.spectrogram(y, nfft=256, fs=256, return_onesided=True, noverlap=128)[2]
    Pxx = np.delete(Pxx, np.s_[117:123+1], axis=0)
    Pxx = np.delete(Pxx, np.s_[57:63+1], axis=0)
    Pxx = np.delete(Pxx, 0, axis=0)

    result=(10*np.log10(np.transpose(Pxx))-(10*np.log10(np.transpose(Pxx))).min())/(10*np.log10(np.transpose(Pxx))).ptp()
    return result
```

```
        def createSpectrogram(data, S=0):
            global nSpectogram
            global signalsBlock
            global inB
            signals=np.zeros((22,59,114))

            t=0
            movement=int(S*256)
            if(S==0):
                movement=_SIZE_WINDOW_SPECTOGRAM
            while data.shape[1]-(t*movement+_SIZE_WINDOW_SPECTOGRAM) > 0:
                # CREAZIONE DELLO SPETROGRAMMA PER TUTTI I CANALI
                for i in range(0, 22):
                    start = t*movement
                    stop = start+_SIZE_WINDOW_SPECTOGRAM
                    signals[i,:]=createSpec(data[i,start:stop])
                if(signalsBlock is None):
                    signalsBlock=np.array([signals])
                else:
                    signalsBlock=np.append(signalsBlock, [signals], axis=0)
                nSpectogram=nSpectogram+1
                if(signalsBlock.shape[0]==50):
                    saveSignalsOnDisk(signalsBlock, nSpectogram)
                    signalsBlock=None
                    # SALVATAGGIO DI SIGNALS
                t = t+1
            return (data.shape[1]-t*_SIZE_WINDOW_SPECTOGRAM)*-1
```

Figura 3.1: "createSpec" e "createSpectogram"

Prima di andare a definire il fattore **S**, definito sopra (in 2.2) devo andare a dividere e salvare in memoria dei dati interessanti sulle ore di inizio fine dei segnali pre-ictali o interictali. Questi si trovano sui file "summary" all'interno della cartella di ogni paziente.

Lo spostamento **S** è definito come:
$\mathbf{S} = (SizeWindowSpect./256 * (len(preictalinfo) * 30 min.ofdata/256))/(tottInst)$

Ricordiamo che tale metodo ci permette di creare dati sintetici coerenti (e non "inventati" o "arealistici").

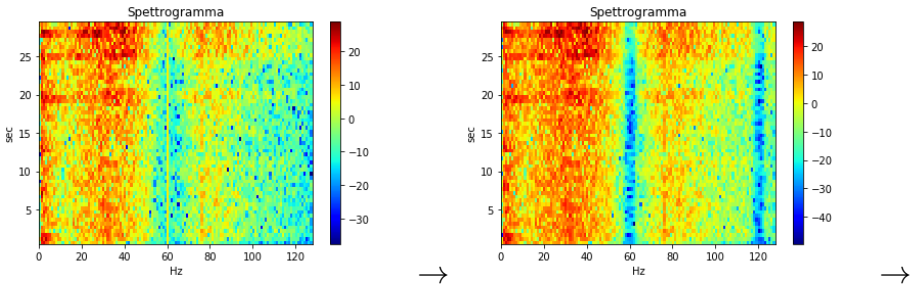


Figura 3.2: nella figura a sinistra c'è uno spettrogramma originale. Nella figura a destra si attua la modulazione per le frequenze a 0 Hz e per gli intervalli di frequenza a $57 - 63$ Hz e $117 - 123$ Hz.

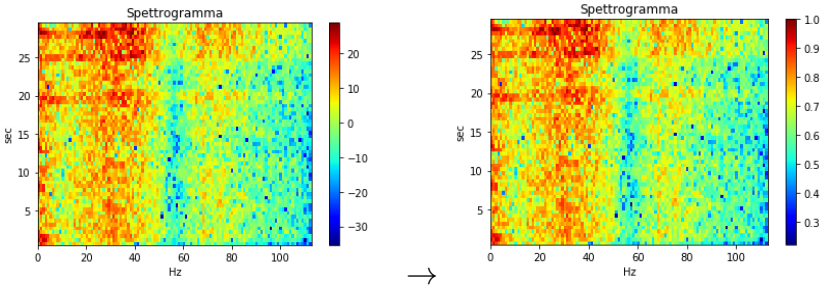


Figura 3.3: Nella figura a sinistra vengono tolte le frequenze che erano state modulate allo step prima. Nella figura di destra vediamo lo spettrogramma di output pronto per entrare nella rete.

3.3 CNN

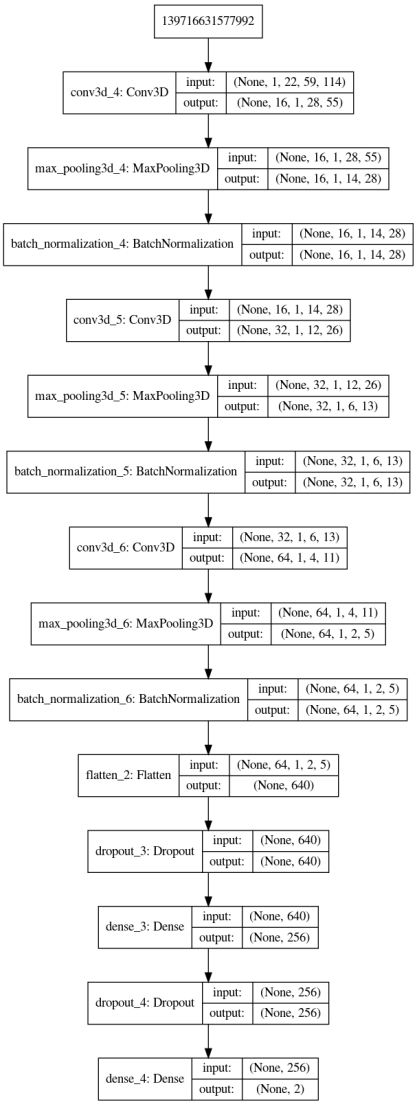
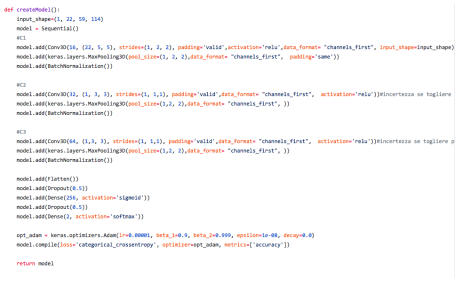


Figura 3.4: Architettura della CNN proposta

Il codice servito per tale fine è quello che si può leggere nel file "CNN.py". Nel codice viene utilizzata la libreria *Keras*. In primis viene creato con "createModel():" il modello della rete a tre blocchi di convoluzione (come descritto sopra).



```
def createModel():
    input_shape=(1, 22, 59, 114)
    model = Sequential()
    #C1
    model.add(Conv3D(16, (22, 5, 5), strides=(1, 2, 2), padding='valid',activation='relu',data_format= "channels_first", input_shape=input_shape))
    model.add(keras.layers.MaxPooling3D(pool_size=(1, 2, 2),data_format= "channels_first",  padding='same'))
    model.add(BatchNormalization())

    #C2
    model.add(Conv3D(32, (1, 3, 3), strides=(1, 1,1), padding='valid',data_format= "channels_first",  activation='relu'))#incertezza se togliere
    model.add(keras.layers.MaxPooling3D(pool_size=(1,2, 2),data_format= "channels_first", ))
    model.add(BatchNormalization())

    #C3
    model.add(Conv3D(64, (1,3, 3), strides=(1, 1,1), padding='valid',data_format= "channels_first",  activation='relu'))#incertezza se togliere p
    model.add(keras.layers.MaxPooling3D(pool_size=(1,2, 2),data_format= "channels_first", ))
    model.add(BatchNormalization())

    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(256, activation='sigmoid'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))

    opt_adam = keras.optimizers.Adam(lr=0.00001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
    model.compile(loss='categorical_crossentropy', optimizer=opt_adam, metrics=['accuracy'])

    return model
```

Figura 3.5: "createModel"

Con le funzioni "generatearraysfortraining" e "generatearraysforpredict" si definisce come la rete deve caricare i dati e come vanno usati per il training e per il testing.

```
def generate_arrays_for_training(indexPat, paths, start=0, end=100):
    while True:
        from_=int(len(paths)/100*start)
        to_=int(len(paths)/100*end)
        for i in range(from_, int(to_)):
            f=paths[i]
            x = np.load(PathSpectogramFolder+f)
            x=np.array([x])
            x=x.swapaxes(0,1)
            if('P' in f):
                y = np.repeat([[0,1]],x.shape[0], axis=0)
            else:
                y =np.repeat([[1,0]],x.shape[0], axis=0)
            yield(x,y)

def generate_arrays_for_predict(indexPat, paths, start=0, end=100):
    while True:
        from_=int(len(paths)/100*start)
        to_=int(len(paths)/100*end)
        for i in range(from_, int(to_)):
            f=paths[i]
            x = np.load(PathSpectogramFolder+f)
            x=np.array([x])
            x=x.swapaxes(0,1)
            yield(x)
```

Figura 3.6: "generatearraysfortraining" e "generatearraysforpredict"

La parte più importante è la funzione "main", nella quale avviene il lavoro della CNN, utilizzando il metodo della Leave-One-Out cross-validation.

Specifichiamo che:

- durante l'allenamento i dati non vengono caricati tutti insieme, in una sola volta, ma il modello si preoccupa di caricare tali dati un po' alla volta. Questa scelta è stata fatta per motivi pratici di memoria;

```
model.fit_generator(generate_arrays_for_training(indexPat, filesPath, end=75), #end=75),#It take the first 75%
                    validation_data=generate_arrays_for_training(indexPat, filesPath, start=75),#start=75), #It take the l
                    #steps_per_epoch=10000, epochs=10)
                    steps_per_epoch=int((len(filesPath)-int(len(filesPath)/100*25))),#*25),
                    validation_steps=int((len(filesPath)-int(len(filesPath)/100*75))),#*75),
                    verbose=2,
                    epochs=300, max_queue_size=2, shuffle=True, callbacks=[callback])# 100 epochs è meglio #aggiungere cri
```

- la rete non procede se si raggiunge un valore di validation accuracy superiore del 90%

```
callback=EarlyStoppingByLossVal(monitor='val_acc', value=0.975, verbose=1, lower=False)
```

Il codice "TestThreshold.py" è stato creato per ottimizzare, per ogni paziente, il valore soglia (output di CCN.py), al fine di ottenere valori FPR e di sensitività sempre migliori.

3.4 Risultati

I risultati ottenuti sono questi:

Paziente	N° attacchi	Ore intericta l		Sensitività (%)		FPR (/h)	
		Articolo	Progetto	Articolo	Progetto	Articolo	Progetto
Paz1	7	17	14,49	85,7±0,0	84,75±30,88	0,24±0,00	0,54±1,79
Paz2	3	22,9	26,26	33,3±0,0	41,42±46,98	0,00±0,00	1,80±2,86
Paz5	5	13	14,57	80±20	77,43±39,28	0,19±0,00	3,05±3,34
Paz19	3(2)	24,9	26,21	100±0,0	84,61±30,76	0,00±0,00	11,98±5,92
Paz21	4	20,9	23,63	100±0,0	79,82±32,95	0,23±0,09	1,67±2,32
Paz23	5	12,5	14,25	100±0,0	84,82±27,09	0,33±0,00	1,74±2,92

Come si può leggere, i valori dell'FPR sono leggermente più alti (a causa delle differenze che si hanno nel calcolo delle ore interictali), mentre la Sensitività è paragonabile a quella ricavata nell'artico, sebbene è molto più alta la deviazione standard.

Capitolo 4

Conclusioni

I risultati sono in linea con quelli che si sono avuti nel lavoro [1].

Segnaliamo che gli autori sono stati contattati in quanto, dopo la lettura dell'articolo non erano troppo chiare le procedure seguite dagli autori. A seguito delle varie comunicazioni messe in atto abbiamo realizzato che ci sono state molte omissioni e alcuni errori nel calcolo di dati importanti ai fini della replicazione dei risultati.

Tutto ciò ci ha portati ad avere una visione critica nei confronti dei risultati dell'articolo stesso: sicuramente la CNN è uno strumento utile nei problemi di predizione, ma probabilmente raggiunge dei risultati lievemente inferiori rispetto a quelli riportati.

Bibliografia

[1] Nhan Duy Truong, Ann Duy Nguyen, Levin Kuhlmann, Mohammad Reza Bonyadi, Jiawei Yang, Samuel Ippolito, Omid Kavehei *Convolutional neural networks for seizure prediction using intracranical and scalp electroencephalogram*, Neural Networks, Elsevier 2018.

[2] Yann LeCun, Yoshua Bengio, Geoffrey Hinton *Deep Learning*, Nature, 2015

[3] Aurelien Geron, *Hands-on machine Learning with Scikit-Learn and TensorFlow* O'Reilly, 2017

[4] Yun Park, Lan Luo, Keshab K. Parhi, Theoden Netoff *Seizure prediction with spectral power of EEG using cost-sensitive support vector machines*, Epilepsia, 52(10):1761?1770, 2011

[5] Ali Shoeb, *Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment*, PhD Thesis, Massachusetts Institute of Technology, September 2009.

[6] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE, *PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signal* Circulation 101(23):e215-e220, 2000 (June 13).

Siti consultati

[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.spectrogra

[8] https://pythontic.com/visualization/signals/spectrogram

[9] $https : //www.youtube.com/watch?v = YRhxdVk_sIs\&feature = youtu.be$

[10] https://www.youtube.com/watch?v=cAICT4Al5Ow

[11] $https : //www.youtube.com/watch?v = ZjM_XQa5s6s$

[12] https://keras.io/layers/convolutional/

[13] https://www.youtube.com/watch?v=FTr3n7uBIuE

[14] http://course.fast.ai/lessons/lesson4.html

[15] http://course.fast.ai/lessons/lesson2.html

[16] https://ars.els-cdn.com/content/image/1-s2.0-S235239641730470X-mmc1.pdf

[17] https://www.physionet.org/pn6/chbmit/