Confronto delle performance tra modelli di learning per il riconoscimento delle emozioni tramite segnali EEG

Andrea Amorosini[1,2]

[1]Universita' degli studi di Salerno, Dipartimento di Informatica

Abstract

Questo Paper descrive il processo di riconoscimento delle emozioni tramite l'utilizzo di segnali EEG (elettro-encefalogrammi), i dati sul movimento degli occhi ed una combinazione dei due utilizzando 6 diversi modelli di Machine Learning: SVM , Random Forest, AdaBoost, Gradient Boosting, DNN e CNN. Sono state usate le feature pre-estratte all'interno del dataset SEED-IV per costruire la feature matrix che e' stata poi sottoposta ad un processo di riduzione della dimensionalita' tramite PCA e ad un processo di normalizzazione. I modelli sono poi stati ottimizzati eseguendo un processo di "Hyperparameter Optimization" per ognuno dei modelli esaminati andando poi a validarli facendo uso di tre diversi protocolli di validazione: Subject Dependent, Subject Independent e Subject Biased. I risultati finali vengono poi messi a confronto suddividendoli per ognuno dei tre diversi protocolli e per ognuno dei tre tipi di dati. Andando a rivelare come tra i modelli proposti solo la DNN e la CNN ottengono risultati soddisfacenti conservando la capacita' di generalizzazione.

Introduzione

Il Riconoscimento delle emozioni usando i segnali EEG (elettroencefalogramma) ha ottenuto molta attenzione negli ultimi anni visto il potenziale delle sue applicazioni in vari campi, come la valutazione della salute mentale, l'interazione uomo-macchina, e il calcolo affettivo. [1]

I segnali EEG danno utili informazioni sull'attività' elettrica del cervello, rendendoli una ricca fonte di informazioni per la decodifica degli stati emotivi. In questo articolo di ricerca, viene presentato uno studio comprensivo sul riconoscimento delle emozioni tramite l'uso di segnali EEG, mettendo a confronto le performance di sei diversi modelli di learning: Support Vector Machine (SVM), Random Forest (RF), AdaBoost, Gradient Boosting, Deep Neural Network (DNN), e Convolutional Neural Networks (CNN).

Ognuno di questi modelli ha caratteristiche ed approcci unici al task di riconoscimento delle emozioni. Le SVM sono riconosciute per la loro abilita' nel trattare dati complessi ed ad alta dimensionalita', rendendole adatte per catturare le relazioni intricate nei segnali EEG. Le Random Forest eccellono nel catturare relazioni non lineari, mentre AdaBoost e Gradient Boosting sono metodi di ensemble che migliorano iterativamente dei learner deboli per ottenere delle predizioni accurate. Le DNN sono capaci di imparare rappresentazioni gerarchiche dai segnali EEG, catturando pattern complessi nei dati, mentre le CNN fanno leva sulla loro abilita' di processare dati strutturati a griglia per estrarre caratteristiche rilevanti dagli spettrogrammi dei segnali EEG

Per condurre questa ricerca, viene utilizzato il dataset SEED-IV contenente i dati sui segnali EEG e sul movimento degli occhi di 15 partecipanti in diversi stati emotivi: neutrale, triste, paura e felicita'.

La valutazione di questi modelli verra' condotta considerando lo score di accuratezza di ogni modello attraverso tre diversi protocolli di valutazione: Subject Dependent, Subject independent e Subject Biased. Mettendo a confronto queste metriche si avrà modo di vedere l'efficacia per ogni modello nel classificare accuratamente diversi esempi di stati emotivi determinando la robustezza e la capacita' di generalizzare per ogni modello

Stato Dell'Arte

Nel campo del riconoscimento delle emozioni da segnali EEG, sono state impiegate varie tecniche e modelli per estrarre caratteristiche significative e per classificare gli stati emotivi. Un approccio comunemente usato e' quello dell'analisi spettrale, che consiste nell'analizzare la distribuzione dell'energia tra diverse frequenze di banda nel segnale EEG. E' stata scoperta una correlazione tra le caratteristiche nelle bande come alpha, beta, theta e gamma e specifici stati emotivi. Le caratteristiche nel dominio del tempo come entropia del segnale, complessità del segnale e momenti statistici sono stati impiegati per registrare le dinamiche temporali delle risposte emotive. Tecniche di estrazione delle caratteristiche come la trasformata wavelet e la Independent Component Analysis (ICA) sono risultate efficaci nell'isolare informazioni di rilievo da segnali EEG complessi. Come

modelli di learner ne sono già stati usati alcuni come Support Vector Machine (SVM), k-nearest neighbors (KNN), e modelli di deep learning come Convolutional Neural networks (CNN) e Recurrent Neural Networks (RNN) per task di classificazione. Combinando i modelli e le tecniche di estrazione delle caratteristiche sopra elencati sono stati ottenuti risultati impressivi nell'accuratezza delle predizioni.[1]

Modelli proposti ed ottimizzazioni

I dati nella seguente sezione sono stati ottenuti dalla documentazione ufficiale di Scikit-Learn [2] e da quella di Keras [3].

SVM

La Support Vector Machine (SVM) e' un modello di apprendimento supervisionato comunemente Usato per task di classificazione e regressione, l'algoritmo SVM mira a trovare un iperpiano ottimale che separi i dati appartenenti a diverse classi con il massimo margine. Questo iperpiano viene determinato dai support vectors, che sono i dati più vicini al confine di decisione

Random Forest

Un modello Random Forest e' un metodo di ensemble learning che combina piu' alberi di decisione per calcolare predizioni. Ogni albero di decisione nella foresta e' costruito su un subset casuale dei dati di allenamento e usa un subset casuale di caratteristiche ad ogni split. La casualità nel modello aiuta a ridurre l'overfitting e incrementa l'abilita' di generalizzare del modello.Durante una predizione, ogni albero nella foresta produce indipendentemente una predizione, e la predizione finale viene determinata tramite un processo di votazione o di media.

AdaBoost

AdaBoost (Adaptive Boosting) e' un metodo di apprendimento ensemble che combina piu' weak learner per creare un modello predittivo robusto. Nell'algoritmo AdaBoost, i weak learner (di solito alberi di decisione con profondità limitata o modelli di classificazione deboli) vengono allenati sequenzialmente su versioni pesate dei dati di allenamento. Dopo ogni iterazione, i pesi degli esempi classificati incorrettamente vengono aumentati, forzando i weak learner seguenti a concentrarsi di più su questi esempi. Il modello finale viene costruito aggregando le predizioni di tutti i weak learner, con il contributo di ogni learner pesato in base alla sua accuratezza nell'allenamento.

Gradient Boosting

Gradient Boosting e' una tecnica di Machine Learning che combina piu' weak learner per crearne uno robusto. A differenza di AdaBoost, che si concentra sulla modifica dei pesi degli esempi, Gradient Boosting migliora direttamente i weak learner. L'algoritmo esegue iterativamente il fitting di nuovi weak learner sugli errori residui dei modelli precedenti, usando l'ottimizzazione gradient descent. Ogni weak learner viene allenato per minimizzare la funzione di perdita calcolando il gradiente negativo delle perdita rispetto alle predizioni del modello. I weak learner vengono aggiunti all'ensemble in maniera graduale, ed ogni modello che lo segue corregge gli errori del modello precedente. La predizione finale viene ottenuta aggregando le predizioni di tutti i weak learner, solitamente tramite una somma pesata.

DNN

Una Deep Neural Network (DNN) e' un tipo di rete neurale artificiale caratterizzata dalla sua profondità, che consiste in vari layer di neuroni interconnessi. Le DNN sono progettate per imitare la struttura e il funzionamento del cervello umano, permettendogli di imparare pattern complessi e di fare predizioni accurate. I Layer sono tipicamente organizzati in un layer di input, uno o più layer nascosti, ed un layer di output. Gli hidden layer permettono al network di imparare rappresentazioni gerarchiche dei dati, estraendo progressivamente caratteristiche di livello sempre più alto man mano che le informazioni passano attraverso il network. Le DNN vengono allenate attraverso un processo chiamato backpropagation, dove il network aggiusta i pesi delle sue connessioni per minimizzare la differenza tra le sue predizioni e i valori reali. In questo studio le DNN studiate hanno una struttura con un numero variabile di hidden layer ognuno dei quali seguito da un layer di dropout, e per finire un ultimo layer con 4 canali di output. Il modello verra' addestrato in 40 epoche specificando come funzione di callback una funzione di early-stop basata sulla validation accuracy del modello in ogni dato momento, che andrà a fermare l'addestramento dopo 3 epoche di mancato miglioramento della validation accuracy.

CNN

Una Convolutional Neural Network (CNN) e' un tipo specializzato di DNN particolarmente efficace per processare e analizzare dati strutturati in forma matriciale, come immagini e sequenze. In una CNN, i dati di input vengono processati attraverso una serie di layer convoluzionali, dove ogni layer con-

siste di vari filtri che estraggono caratteristiche rilevanti dall'input. Questi filtri passano sopra i dati di input, eseguendo moltiplicazioni elemento per elemento e aggregando i risultati. Le feature maps risultanti vengono poi passate attraverso funzioni di attivazione non lineari per introdurre non linearità nel modello, Le CNN incorporano anche dei layer di pooling che riducono il campionamento delle feature map, riducendo la loro dimensione spaziale preservando le informazioni più importanti. I layer finali di una CNN sono solitamente composti da layers pienamente connessi, che usano le caratteristiche estratte per fare predizioni o classificazioni. In questo studio le CNN studiate hanno una struttura con numero variabile di layer convoluzionali ognuno dei quali seguito da un layer di Pooling, con infine una struttura composta da un layer di appiattimento dei dati, che verranno fatti passare attraverso un ultimo layer Dense e il suo seguente layer di dropout per finalmente arrivare al suo ultimo layer con 4 canali di output. Come per la DNN il modello verra' addestrato su 40 epoche specificando come funzione di callback sempre una funzione di early-stop basata sulla validation accuracy del modello in ogni dato momento.

Ottimizzazione Scikit-Learn

I Modelli come SVM, Random Forest, AdaBoost e Gradient Boosting hanno tutti degli iperparametri che possono essere ottimizzati per delle performance ottimali. GridSearch, Random Search e Ottimizzazione Bayesian sono metodi popolari per l'ottimizzazione degli iperparametri. Grid Search esegue una ricerca esaustiva da una griglia definita precedentemente come spazio di ricerca degli iperparametri desiderati valutando le performance del modello per ogni combinazione possibile, Random Search selezione iperparametri casualmente da un dato range di valori e valuta le performance del modello per ogni combinazione casuale ed infine l'ottimizzazione Bayesian che impiega un modello probabilistico per predire le performance di varie configurazioni di iperparametri basandosi sulle valutazioni precedenti. Utilizza questa informazione per guidare la ricerca verso regioni promettenti dello spazio degli iperparametri.

In generale si e' notato un incremento minimo del 5% dell'accuracy. Nello specifico, invece, di seguito sono riportate le combinazioni ottimali trovate per ogni modello e l'accuratezza risultante messa a confronto con un modello di default come baseline.

I seguenti parametri sono stati considerati per i 4 modelli sopracitati:

Random Forest

- `n_estimators`: Specifica il numero di alberi di decisione da creare nella random forest
- `max_depth`: Imposta la profondità massima di ogni albero all'interno della random forest
- `max_features`: Determina il numero massimo di caratteristiche da considerare quando si cerca per il miglior split.
- `min_samples_split`: Il numero minimo di esempi richiesti per eseguire lo split di un nodo interno.
- `min_samples_leaf`: Il numero minimo di esempi richiesto esserci su un nodo foglia, uno split verra' eseguito solo se risulta in due nodi che hanno almeno questo numero di esempi al loro interno.
- Bootstrap: Un parametro booleano che indica se utilizzare gli esempi bootstrap quando si costruiscono gli alberi

AdaBoost

- `n_estimators`: Determina il numero massima di weak learners da usare nell'ensemble
- `learning_rate`: Controlla il contributo di ogni weak learner nell'ensemble, scala il peso applicato alle predizioni di ogni weak learner
- `algorithm`: Specifica l'algoritmo di boosting da usare.
- `max_depth`: Profondità massima dell'albero di decisione usato come `base_estimator`

Gradient Boosting

- `n_estimators`: Determina il numero massima di weak learners da usare nell'ensemble
- `learning_rate`: Controlla il contributo di ogni weak learner nell'ensemble, scala il peso applicato alle predizioni di ogni weak learner
- `max_depth`: Imposta la profondità massima di ogni albero di decisione nell'ensemble

SVM

- `C`: Controlla la potenza della regolarizzazione o il trade-off tra il massimizzare i margini e il minimizzare il classification error sui dati di training
- `Gamma`: Definisce il coefficiente del kernel per i kernel "rbf", "poly", e "sigmoid", controlla l'influenza dei singoli esempi di training e la dimensione del decision boundary
- `Kernel`: Specifica il tipo di funzione kernel usata per trasformare i dati di input in uno spazio delle caratteristiche a dimensionalita' più alta

Vengono di seguito presentati sotto forma di tabelle gli esperimenti e i loro risultati per l'ottimizzazione degli iperparametri dei modelli facendo riferimento ad una validazione Subject-Biased

Table 1: Ricerca effettuata e risultati ottenuti per ogni modello con scikit-learn

Modello	Spazio di ricerca	Iperparametri ottimali	Accuracy modello ottimizzato	Accuracy modello baseline	Modello baseline
Random Forest	n_estimators : [20-2000] max_depth : [10-310] max_features: [Sqrt] min_samples_split: [1-60] min_samples_leaf: [1-10] Bootstrap: [True, False]	n_estimators: 530, max_depth: 250, max_features: Sqrt, min_samples_split: 20, min_samples_leaf: 2, Bootstrap: True	0.694444	0.611111	n_estimators: 100, max_depth: None, max_features: Sqrt, min_samples_split: 2, min_samples_leaf: 1, Bootstrap: True
AdaBoost	n_estimators: [200-2000] learning_rate: [0.0001, 0.001, 0.01, 0.1, 1.0] Algorithm: [“SAMME”, “SAMME.R”] max_depth: [1-10]	n_estimators: 360, learning_rate: 0.001, Algorithm: “SAMME.R”, max_depth: 5	0.652778	0.467593	n_estimators: 50, learning_rate: 1, Algorithm: “SAMME.R”, max_depth: 1
Gradient Boosting	n_estimators: [100-500] learning_rate: [0.0001-1.0] max_depth: [2-10]	n_estimators: 150, learning_rate: 0.05, max_depth: 2	0.643518	0.597222	n_estimators: 100, learning_rate: 0.1, max_depth: 3
SVM	C: [0.1, 1, 10, 100] Gamma: [scale, auto] Kernel: [rbf, linear, poly]	C: 0.1, Gamma : scale, Kernel: linear	0.666667	0.638889	C : 1, Gamma : scale, Kernel : rbf

Ottimizzazione KerasTuner

Il Processo di ottimizzazione utilizzando Keras Tuner permette la ricerca automatica degli iperparametri migliori per un modello di deep learning. Utilizzando anche i metodi descritti nel paragrafo precedente Keras tuner esegue una ricerca basandosi su una funzione modello descritta ad hoc per ottimizzare parametri come il numero di layer, il numero di filtri, la presenza o meno di layer di pooling e la regola di regolarizzazione scelta per ogni layer. La regolarizzazione consiste nello specificare quale regola utilizzare per ogni layer scegliendo tra l1, l2 o `l1_l2`, che andranno ad introdurre dei termini di penalità per la funzione di loss, incoraggiando il modello a trovare un punto di equilibrio tra il fitting dei dati di allenamento ed il mantenere i pesi delle caratteristiche ridotti. Mettendo questa restrizione sui pesi delle caratteristiche, la regolarizzazione aiuta a ridurre la complessità del modello, prevenendo eventuali casi di overfitting.

Viene qui presentata la combinazione ottimale di iperparametri per la DNN e la CNN con annessa accuratezza del modello risultante messa a confronto con un modello semplice usato come baseline.

I seguenti parametri sono stati considerati per i 2 modelli sopracitati:

DNN

- `num_layers`: Numero di layer contenuti all'interno della rete neurale
- `units_i` : Corrispondente al parametro units dell'i-esimo layer, specifica il numero di neuroni nel layer Dense e determina la dimensionalita' dell'output del layer
- `regularization_i` : Corrisponde al parametro `kernel_regularizer` dell'iesimo layer, permette di applicare una regolarizzazione ai pesi kernel del layer Dense, questa regolarizzazione aiuta a prevenire l'overfitting aggiungendo un termine di penalità alla funzione di loss.
- `dropout_i` : Corrisponde al parametro rate del layer di Dropout che viene subito dopo l'i-esimo layer della rete neurale, specifica il rateo di Dropout ovvero quanto delle unita' di input dovrà essere modificato in 0 durante il training.
- `learning_rate` : Parametro `learning_rate` dell'algoritmo di ottimizzazione Adam utilizzato, e determina la magnitudine degli aggiornamenti ai pesi del modello durante il training

CNN

- `conv_layers`: Numero di layer convoluzionali all'interno della rete convoluzionale.
- `filters_i`: Corrispondente al parametro filters per l'i-esimo layer, specifica il numero di filtri o kernel da usare nel layer, ogni filtro rappresenta un feature detector che viene applicato sulla sequenza di input, inoltre determina il numero di canali di output del layer.
- `kernel_size_i`: Corrispondente al parametro `kernel_size` per l'i-esimo layer, definisce la dimensione del filtro e rappresenta la lunghezza della finestra convoluzionale.
- `max_pooling_i`: Corrispondente al parametro `pool_size` del layer di pooling che segue l'i-esimo layer, presente per ridurre la dimensione spaziale dei dati e ridurre la presenza di overfitting, e specifica la dimensione della finestra di pooling
- `regularizer_i`: Corrispondente al parametro `kernel_regularizer` dell'i-esimo layer, permette l'applicazione di regolarizzazione ai pesi del kernel del layer convoluzionale cosi' da prevenire overfitting
- `regularizer_end`: Corrispondente al parametro `kernel_regularizer` del layer Dense finale.
- `units_end`: Corrispondente al parametro units per il layer Dense finale.
- `learning_rate`: Parametro `learning_rate` dell'algoritmo di ottimizzazione Adam utilizzato, e determina la magnitudine degli aggiornamenti ai pesi del modello durante il training

Vengono di seguito presentati sotto forma di tabelle gli esperimenti e i loro risultati per l'ottimizzazione degli iperparametri dei modelli facendo riferimento ad una validazione Subject-Biased

Table 2: Ricerca effettuata e risultati ottenuti per ogni modello con KerasTuner

Modello	Spazio di ricerca	Iperparametri ottimali	Accuracy modello ottimizzato	Accuracy modello baseline	Modello baseline
DNN	num_layers: [2-5], units_i: [150-512], regularization_i: [“l1”,”l2”,”l1_l2”], dropout_i: [0-0.5], learning_rate: [1e-2, 1e-3, 1e-4]	num_layers: 2, units_1: 406, regularization_1: “l1_l2”, dropout_1: 0.05, units_2: 310, regularization_2: “L2”, dropout_2: 0.29, learning_rate: 0.01	0.634259	0.578704	num_layers: 4, units_1: 214, regularization_1:“l1”, dropout_1: 0.3, units_2: 214, regularization_2: “L2”, dropout_2: 0.1, units_3: 342, regularization_3: “L2”, dropout_3: 0.2, units_4: 374, regularization_4: “L1”, dropout_4: 0.3, learning_rate; 0.0001
CNN	conv_layers: [1-4], filters_i: [4-64], kernel_size_i: [2-5], max_pooling_i: [“max”, “avg”[, regularizer_i: [“l1”, “l2”, “l1_l2”], regularizer_end: [“l1”, “l2”, “l1_l2”], units_end: [70-150], learning_rate: [1e-2, 1e-3, 1e-4]	n_estimators: 360, learning_rate: 0.001, Algorithm:“SAMME.R”, max_depth: 5	0.638889	0.62963	conv_layers: 1, filters_1: 32, kernel_size_1: 3, max_pooling_1: None, regularizer_1: None, regularizer_end: None, units_end: 70, learning_rate: 0.001

Metodologie

Dataset

Il Dataset selezionato per questo studio e' il SEED-IV [4] che e' composto da dati sui segnali EEG e sul movimento degli occhi registrati da 15 soggetti utilizzando un lettore a 62 canali, contenendo per ogni soggetto 3 sessioni composte da 24 trial ognuna.

Ogni trial ha una label corrispondente tra: neutrale, tristezza, paura, e felicita'. Per questo studio vengono utilizzate le caratteristiche estratte dai segnali EEG già presenti nel dataset per le quali al segnale EEG e' stato prima applicato un downsampling dai 1000Hz originali a 200Hz, poi e' stato filtrato con un filtro passa-banda tra 1Hz e 75Hz ed infine sono state estratte le caratteristiche; la Power Spectral Density (PSD) e la differential Entropy (DE) suddivise in 5 bande: delta (1-4Hz), theta (4-8Hz), alpha (8-14Hz), beta (14-31Hz) e gamma (31-50Hz). Inoltre vengono prese in considerazione anche le caratteristiche estratte dal movimento degli occhi registrato durante le varie trial, tra queste abbiamo dati sul diametro delle pupille, sulla dispersione, sulla durata del battito di ciglia, sulla saccade e altri dati statistici degli occhi. Per ulteriori informazioni sulle tecniche di preprocessing utilizzate e le feature estratte si faccia riferimento a [4]

Eye movement parameters	**Extracted features**
Pupil diameter (X and Y)	Mean, standard deviation and DE features in four bands: 0-0.2 Hz, 0.2-0.4 Hz, 0.4-0.6 Hz, and 0.6-1 Hz
Dispersion (X and Y)	Mean, standard deviation
Fixation duration (ms)	Mean, standard deviation
Blink duration (ms)	Mean, standard deviation
Saccade	Mean, standard deviation of saccade duration (ms) and saccade amplitude (°)
Event statistics	Blink frequency, fixation frequency, maximum fixation duration, total fixation dispersion, maximum fixation dispersion, saccade frequency, average saccade duration, average saccade amplitude, and average saccade latency.

Figure 1: Lista delle caratteristiche estratte dal movimento degli occhi

Riduzione e normalizzazione della feature matrix

La riduzione delle feature matrix usando la Principal Component Analysis (PCA) e' una tecnica comunemente usata per ridurre la dimensionalita' dei dati mantenendo le informazioni importanti. Il processo della PCA include vari step. Primo, la feature matrix viene standardizzata sottraendo la media e scalando ogni feature per avere una varianza unitaria. Questo step fa in modo tale che tutte le feature abbiano un eguale influenza durante il processo di riduzione della dimensionalita'. Poi viene calcolato la matrice di covarianza della feature matrix normalizzata, questa quantifica la relazione tra le diverse feature. PCA poi identifica i componenti principali, che sono le combinazioni lineari delle feature originali che catturano la massima varianza nei dati. Queste componenti principali sono ortogonali fra di loro e vengono classificate in ordine dell'ammontare di varianza che rappresentano. Finalmente, un subset delle componenti classificate meglio viene selezionato in base all'ammontare desiderato di varianza da mantenere. La feature matrix viene poi trasformata proiettandola su queste componenti principali selezionate, risultando in una feature matrix di dimensionalita' ridotta.

La normalizzazione, invece, e' una tecnica comunemente usata per prevenire l'overfitting nei modelli addestrati e per migliorare la generalizzazione di questi. Quando la normalizzazione viene applicata ad una feature matrix con valori tra 0 ed 1, tipicamente effettua un processo di scaling ai valori all'interno di uno specifico range. Questo processo assicura che tutte le feature abbiano eguale influenza durante il training del modello e previene che certe feature possano dominare il processo di learning per via della loro maggior magnitudine. Conseguentemente, La normalizzazione a valori tra 0 e 1 in una feature matrix aiuta nell'ottenere un modello più stabile e più performante.

Metodi di validazione

Si e' deciso di seguire i protocolli per la validazione usati in [5], la seguente e' una breve spiegazione del funzionamento dei tre protocolli scelti per questo studio:

Subject-Dependent
Le impostazioni dell'esperimento adottate consistono in 16 trial che vengono scelte casualmente come training set e le rimanenti 8 che vengono usate per il testing. In ogni allocazione casuale, ci si assicura che ci siano due esempi per ogni categoria di label all'interno del validation set per permettere di avere un equilibrio di classi per questo dataset relativamente piccolo. Il risultato finale vera' poi determinato facendo la media tra tutti i soggetti e le sessioni. Questo metodo aiuta a determinare la performance del modello su dati familiari ma non rispec-

chia adeguatamente la sua performance su dati mai visti.

Subject-Independent
Si esegue la leave-one-out cross-validation (LOOCV) per tutti i soggetti. Questa validazione verra' eseguita per ogni sessione, dove per ogni soggetto questo verra' usato come validation set e verranno usati i 14 soggetti rimanenti come training set. Il risultato finale verra' poi calcolato dalla media dei risultati delle tre sessioni. Questo metodo constata l'abilita' del modello sul generalizzare su nuovi soggetti e fornisce una stima più realistica delle performance del modello.

Subject-Biased
Vengono illustrati gli effetti del biased sampling, dove una porzione casuale dei dati in una particolare trial vengono usati nel training e i rimanenti vengono usati per il testing. Ad esempio, 20% dei dati da ogni soggetto nella stessa trial vengono casualmente scelti per il testing mentre il rimanente viene usato per il training. Nello specifico, vengono considerati i dati di tutti i soggetti sia per il training che per il testing senza separarli in dati individuali per ogni soggetto come fatto nei protocolli precedenti. Questo metodo, infine, aiuta ad identificare potenziali limitazioni o bias nelle predizioni del modello, favorendo cosi' ulteriori miglioramenti.

Risultati

Vengono ora presentati i risultati dello studio, mettendo a confronto la misura di accuratezza per ognuno dei modelli, in ognuno dei tre protocolli di validazione. Inoltre sono presentate le misure di accuratezza dei modelli con i dati sul movimento oculare e con l'unione di questi ultimi con i dati sui segnali EEG.

Table 3: Accuratezza dei modelli per i dati dei segnali EEG

Modello	Subject Dependent	Subject Independent	Subject Biased
SVM	0.258333	0.769444	0.666667
Random Forest	0.272222	0.810185	0.694444
AdaBoost	0.352778	0.672222	0.652778
Gradient Boosting	0.286111	0.777778	0.643518
DNN	0.25	0.802778	0.62037
CNN	0.330556	0.748148	0.638889

Table 4: Accuratezza dei modelli per i dati sul movimento oculare

Modello	Subject Dependent	Subject Independent	Subject Biased
SVM	0.283333	0.857407	0.819444
Random Forest	0.313889	0.880556	0.837963
AdaBoost	0.222222	0.758333	0.592593
Gradient Boosting	0.286111	0.87037	0.796296
DNN	0.261111	0.862037	0.759259
CNN	0.263889	0.85463	0.782407

Table 5: Accuratezza dei modelli per i dati sui segnali EEG combinati con i dati sul movimento oculare

Modello	Subject Dependent	Subject Independent	Subject Biased
SVM	0.183333	0.803704	0.768519
Random Forest	0.277778	0.847222	0.740741
AdaBoost	0.283333	0.669444	0.458333
Gradient Boosting	0.2252	0.817593	0.717593
DNN	0.180556	0.824074	0.722222
CNN	0.316667	0.780556	0.740741

Vengono di seguito analizzati i risultati ottenuti da ognuno dei tre metodi di validazione:

Subject-Dependent
Dall'analisi notiamo come con tutti i tre tipi di dati la Subject-dependent non ha riportato risultati soddisfacenti raramente superando un accuracy dello 0.30, rimanendo su una media dello 0.26, ed inoltre dimostrando peggioramenti allo score quando presentato con i due tipi di dati combinati.

Subject-Independent
Possiamo notare come, in media, l'aggiunta dei dati sul movimento oculare porti in generale ad un miglioramento dello score della accuracy tra lo 0.2 e lo 0.4, non riuscendo pero' a superare i punteggi della validazione solo sui dati sul movimento oculare. Inoltre tra i 6 modelli proposti possiamo vedere come generalmente restituiscano uno score migliore modelli come l'SVM e la Random Forest, con CNN e DNN che seguono subito dopo e mostrando come AdaBoost non riesca a raggiungere i punteggi raggiunti dagli altri modelli.

Subject-Biased
Generalmente, nella validazione subject-biased possiamo notare un andamento simile a quello presentato nella Subject-Independent solo con un peggioramento sullo score dell'accuracy dello 0.7 (+/- 0.3).

Inoltre e' stato osservato che modelli come Random forest, AdaBoost, e Gradient Boosting nonostante abbiano misure simili agli altri modelli presentino un forte overfitting sui dati, cosi' come mostrato dalle

seguenti curve di apprendimento.

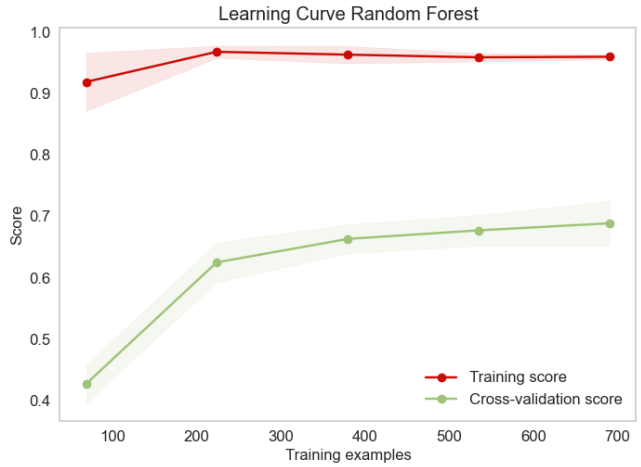

Figure 2: Learning Curve Random Forest

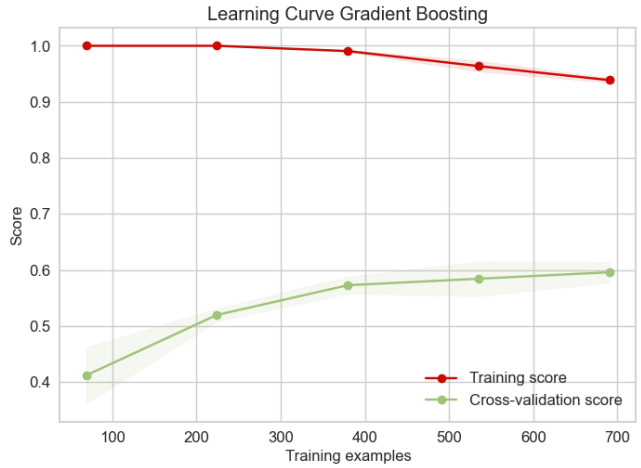

Figure 3: Learning Curve Gradient Boosting

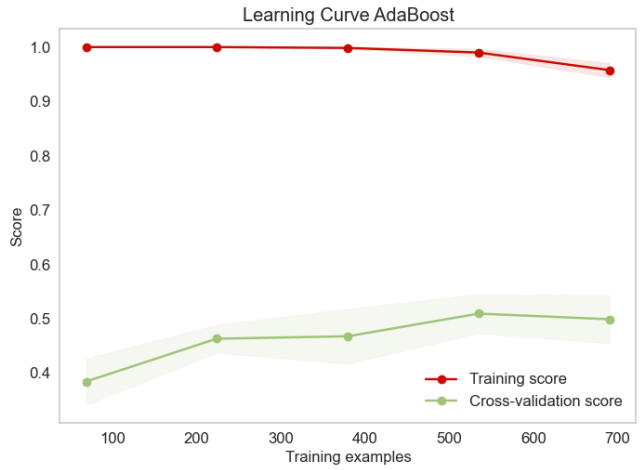

Figure 4: Learning Curve AdaBoost

Come si può' vedere, ognuno dei tre modelli presenta una grande disparita' tra lo score della fase di training e lo score della fase di valutazione. Dopo aver provato ancora con un tuning degli iperparametri, vedendo la mancanza di miglioramenti e' stato assodato che questi tre modelli non sono adatti al task di riconoscimento delle emozioni non riuscendo a generalizzare abbastanza.
Di contro, invece, il modello SVM sembra soffrire di underfitting e quindi non riuscire a comprendere appieno la relazione tra i dati di input e le categorie presentate, cosi' come si puo' vedere nella sua curva di apprendimento

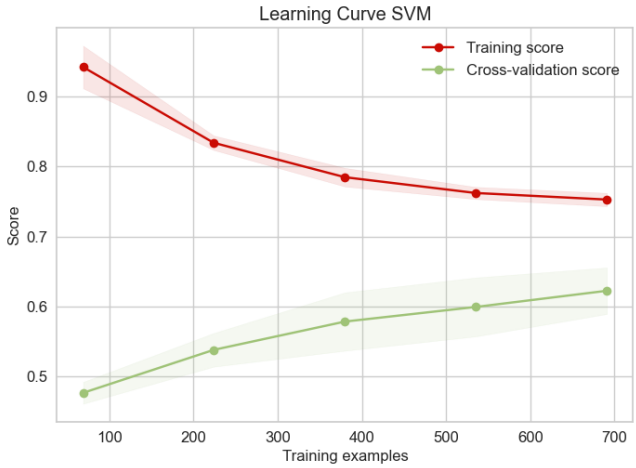

Figure 5: Learning Curve SVM

Infine, i modelli che sono risultati migliori sia come punteggi che come capacita' di generalizzazione sono stati DNN e CNN, che sono riusciti a riportare risultati soddisfacenti senza andare ne' in overfitting ne' in underfitting cosi' come si può vedere nelle loro curve di apprendimento.

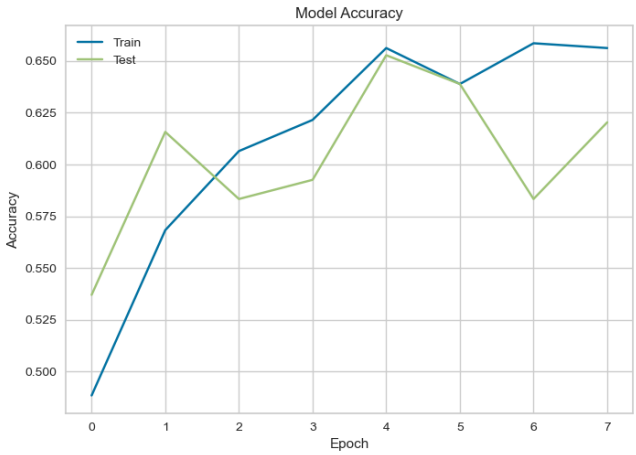

Figure 6: Learning Curve DNN

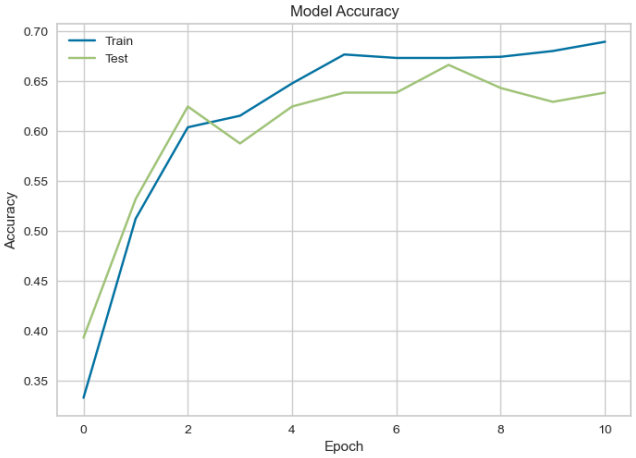

Figure 7: Learning Curve CNN

Possiamo infatti notare come le due curve seguano lo stesso andamento di miglioramento, con la CNN che rispecchia questa caratteristica meglio di qualunque altro modello esaminato.

Conclusioni

Questa ricerca ha rivelato la direzione migliore da intraprendere per studi futuri sul riconoscimento delle emozioni, infatti e' stato osservato come dei sei modelli quelli a restituire i risultati migliori sono stati SVM , DNN e CNN che hanno mostrato avere una buona accuratezza senza mostrare segni di overfitting o underfitting con i tre tipi di dati con cui sono stati eseguiti gli esperimenti (segnali EEG, movimento oculare, ed entrambi) , mentre i rimanenti, ovvero Adaboost, Gradient Boosting e Random Forest hanno mostrato chiari segni di overfitting a qualunque tipo di dato provato. E' stata osservata, inoltre, un ottima performance da parte dei modelli quando allenati sui dati del movimento oculare, andando a sorpassare i punteggi sia dei segnali EEG che dell'unione dei due.

References

[1] Min Wang et al. "Representation Learning and Pattern Recognition in Cognitive Biometrics: A Survey". In: *Sensors* 22.14 (2022). ISSN: 1424-8220. DOI: 10.3390/s22145111. URL: https://www.mdpi.com/1424-8220/22/14/5111.

[2] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[3] François Chollet et al. *Keras*. https://keras.io. 2015.

[4] W. Zheng et al. "EmotionMeter: A Multimodal Framework for Recognizing Human Emotions". In: *IEEE Transactions on Cybernetics* (2018), pp. 1–13. ISSN: 2168-2267. DOI: 10.1109/TCYB.2018.2797176.

[5] Wai-Cheong Lew et al. "EEG-based Emotion Recognition Using Spatial-Temporal Representation via Bi-GRU". In: vol. 2020. July 2020, pp. 116–119. DOI: 10.1109/EMBC44109.2020.9176682. URL: https://www.researchgate.net/publication/343943178_EEG-based_Emotion_Recognition_Using_Spatial-Temporal_Representation_via_Bi-GRU.