Progetto: Age from brain Data

Marco Accerenzi

Esame del 12 Settembre 2024

1 Introduzione e struttura del codice

L'obiettivo del progetto è stato sviluppare un modello di regressione di deep learning per predire l'età cerebrale dei pazienti dai dati di risonanza magnetica (MRI). Il modello raggiunge un errore quadratico medio (RMS) di circa 6 anni. La configurazione ottimale è costituita da un modello con 2 hidden layers e 6 nodi per strato, ottenendo le migliori performance su dati standardizzati e normalizzati.

Sono state utilizzate le seguenti librerie:

Keras per la costruzione del modello di regressione, *Pandas* per la lettura e manipolazione dei dati forniti in formato Excel, *Matplotlib* per tracciare e visualizzare i risultati, *os* e *sys* per la gestione dei percorsi e directory. Il codice si compone di classi e script per la gestione completa del flusso dati e la modellazione:

ExcelData: legge il file Excel e prepara i dati per il modello, **RegressionModel**: implementa il modello e i metodi per addestrare, salvare e caricare il modello, **Console**: offre un'interfaccia utente semplice per l'addestramento e la predizione, **Optimizator**: permette la scelta rapida degli iperparametri, **Analysis**: automatizza l'analisi dei risultati delle predizioni.

2 Estrazione e manipolazione dei dati

I dati MRI sono forniti in due file Excel con 915 campioni. Ogni campione contiene informazioni sull'età reale e il gruppo di provenienza dei dati. La classe *ExcelData* utilizza *Pandas* per caricare i dati in un dataframe e poi li converte in array *numpy* per l'input nel modello.

Prima dell'addestramento, i dati vengono mescolati per prevenire l'overfitting e i valori mancanti vengono riempiti con il valore -9999. Successivamente, i dati vengono normalizzati, ovvero ogni valore è diviso per il massimo della sua colonna. Le celle con valore -9999 vengono escluse dai calcoli. La normalizzazione e la standardizzazione riducono i tempi di addestramento e migliorano la precisione delle previsioni, come mostrato in Figure 1 e Figure 2.

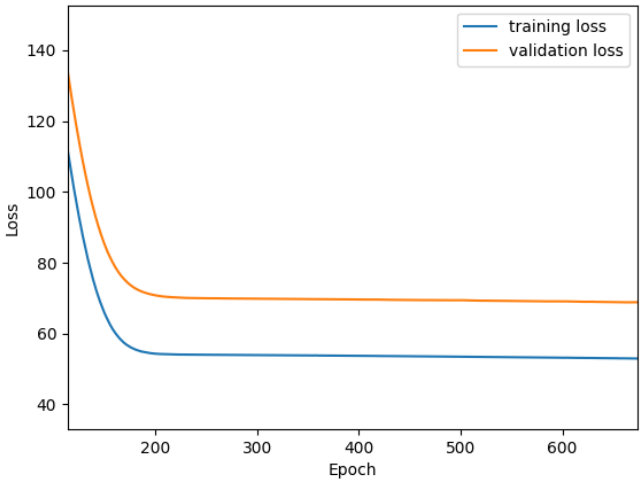

Figure 1: Grafico delle prime epoche di addestramento utilizzando dati normalizzati.

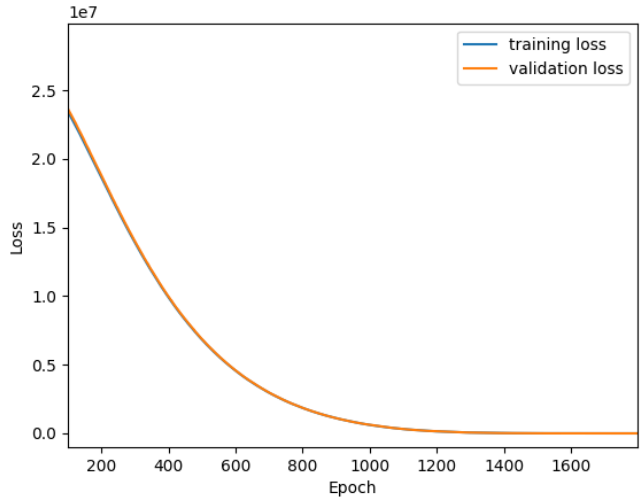

Figure 2: Grafico delle prime epoche di addestramento utilizzando dati grezzi.

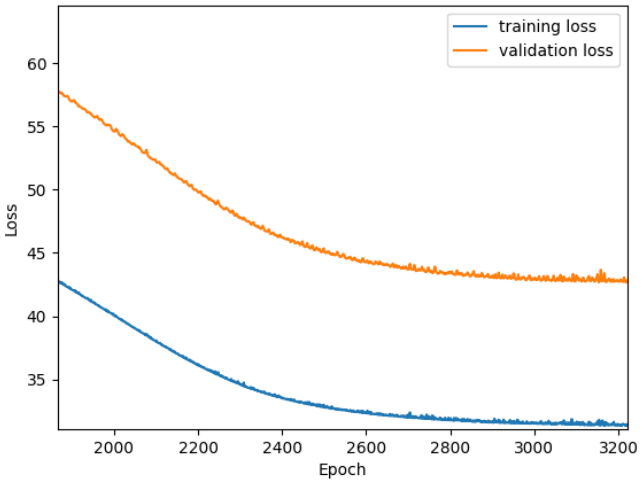

Figure 3: Grafico delle ultime epoche di addestramento utilizzando dati normalizzati.

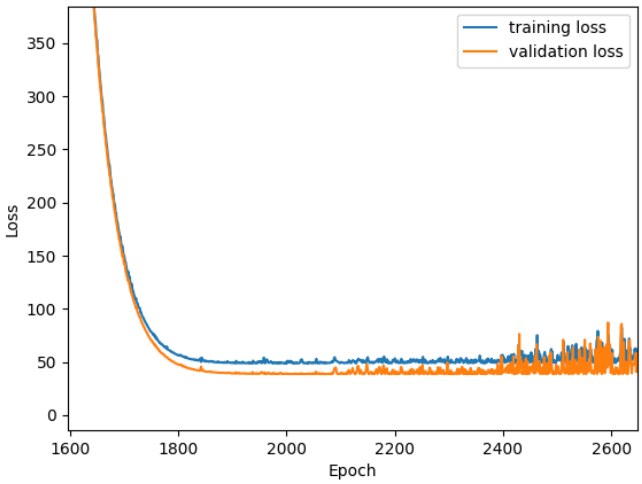

Figure 4: Grafico delle ultime epoche di addestramento utilizzando dati grezzi.

3 Il regression model

Il modello di regressione utilizza i moduli *models*, *layers* e *callbacks* di *keras*. Tra i metodi implementati:

Compile_Model: definisce e compila il modello, **Start_Training**: addestra il modello utilizzando l'algoritmo di apprendimento supervisionato, **Plot_History** e **Save_Model**: salvano rispettivamente la cronologia dell'addestramento e il modello stesso, **load_model**: permette di caricare un modello già addestrato per l'inferenza. Per migliorare l'efficacia dell'addestramento, vengono utilizzati i seguenti callback:

Early stopping: interrompe l'addestramento se la funzione di perdita non migliora dopo un certo numero di epoche, prevenendo l'overfitting, **ModelCheckpoint**: salva checkpoint durante l'addestramento per evitare la perdita di progressi, **ReduceLROnPlateau**: riduce il learning rate quando il modello raggiunge un plateau, **PrintProgress**: stampa il progresso ogni n epoche.

3.1 Ottimizzazione degli iperparametri

Lo script *Optimizator.py* è stato utilizzato per scegliere gli iperparametri. Esplora diversi valori ragionevoli degli iperparametri e realizza un addestramento breve per permettere all'utente di scegliere la configurazione ottimale. Il miglior modello è risultato avere 2 hidden layers e 6 nodi per strato, con le prestazioni migliori in termini di velocità e accuratezza.

4 Risultati e conclusioni

Il modello addestrato è stato utilizzato per prevedere l'età cerebrale dei pazienti, con un errore RMS costantemente tra 6.3 e 6.9 anni. Il miglior modello ha ottenuto un RMS di 6.30 anni e un MSE di 39.66. Un addestramento ripetuto potrebbe migliorare le prestazioni, ma è improbabile ottenere un RMS inferiore ai 6 anni senza incorrere in overfitting.

Il confronto tra le prestazioni dei diversi gruppi è mostrato in Figure 5, dove si osserva che i gruppi "Caltech", "MaxMun" e "SBL" hanno errori di predizione significativamente superiori alla media. Rimuovere questi outlier dai dati non ha migliorato sostanzialmente le prestazioni complessive del modello.

Appendice

Il progetto utilizza un repository pubblico su GitHub.

Ha un ramo principale e tre rami attivi: develop, feature/docs e feature/presentation.

La documentazione del progetto è stata generata tramite Sphinx, utilizzando la sua funzionalità *autodoc* per estrarre le docstring dal codice sorgente.

Il repository pubblico su GitHub consente di distribuire la documentazione del progetto attraverso ReadTheDocs e le Pages di GitHub.

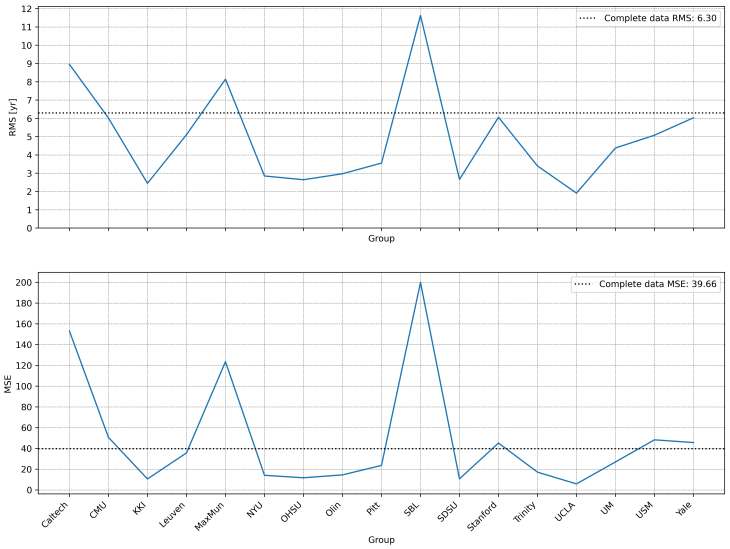

Figure 5: Grafico che confronta l'errore di previsione per i vari gruppi di dati. L'errore sull'intero set di dati è mostrato come una linea tratteggiata.

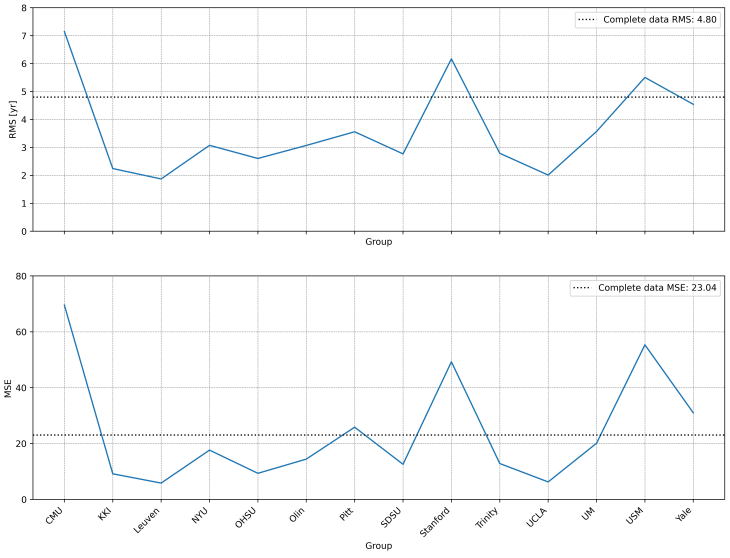

Figure 6: Prestazione del modello allenato senza i tre gruppi a maggiore errore sui soli gruppi a basso errore.

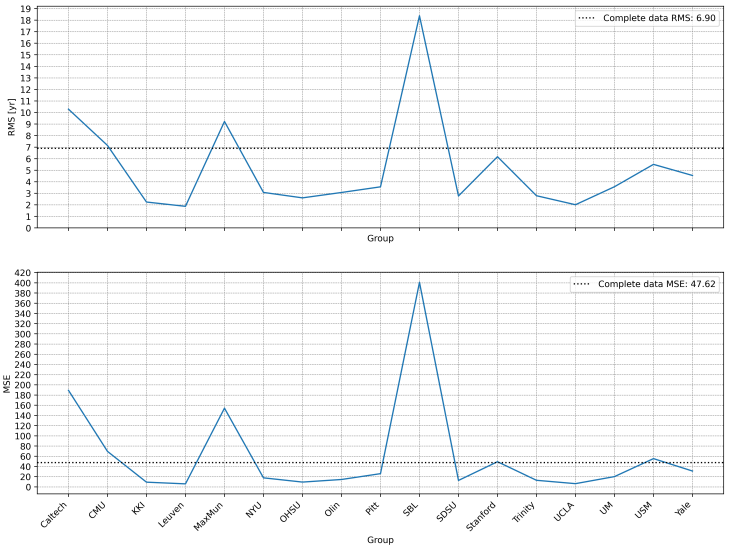

Figure 7: Prestazione del modello allenato con e senza i tre gruppi a maggiore errore.