UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria

Corso di Laurea in Ingegneria Informatica

PROGETTO DI SISTEMI INTELLIGENTI PER INTERNET

Davide Fadale 547045

Oleksandr Poddubnyy 533951

Anno Accademico 2024/2025

Sommario

1. PROBLEMA

Si ha l'obiettivo di suddividere i pazienti in disgrafici e non disgrafici, attraverso la conoscenza di un dataset di immagini di caratteri scritti a mano.

Per il nostro progetto ci serviremo di 2 file, ovvero:

- **Features_extraction**.
- **Classification**.

I risultati sono riportati all'interno di alcuni file in cui vi sono sia rappresentazioni attraverso grafici sia informazioni di natura testuale come la percentuale dei disgrafici e non disgrafici.

Nei capitoli 2 e 3 vengono riportate informazioni di dettaglio in merito ai file Classification e Features_extraction (trattati nel progetto di ML). Entrambi i file sono stati scritti in precedenza per poter andare a svolgere un progetto di Machine Learning; pertanto, abbiamo apportato delle modifiche a questi file per i nostri scopi. In particolare, ci siamo serviti del file Classification per importare il modello di classificazione Random Forest e del file Features_extraction per fare la classificazione delle immagini manoscritte.

2. ESTRAZIONE DELLE FEATURES

Escludendo la sezione di Import, il file "Features_extraction" è suddiviso in:

- Costruzione dataset
- Estrazione delle features con CNN
- Estrazione delle features con OpenCV
- Applicazione della Data Augumentation al dataset originale
- Estrazione delle features con Data Augumentation
- Estrazione di tutte le Features dai modelli pre-addestrati e valutazione dei test

2.1 COSTRUZIONE DEL DATASET

In questa sezione viene caricato il file “users” che contiene i dati dei pazienti. Il file è organizzato in modo tale che ogni riga rappresenti un paziente, caratterizzato da un ID, una label “diag” che indica se è disgrafico o meno, il sesso, la mano con cui scrive e l’età.

Dopodiché è stata creata una funzione encoder che trasforma i valori delle colonne nominali “diag”, “sex” e “hand” in valori numerici binari.

È stata inoltre utilizzata la funzione “get_Session” per accedere, a partire dall’id di uno user, ai dati ricavati a partire dalla sessione che lo user ha svolto per rilevare la disgrafia (ad esempio spazio, pressione, coordinate, ecc).

2.2 ESTRAZIONE DELLE FEATURES CON CNN

Per l’estrazione delle features sono state utilizzate tre architetture deep già implementate, ovvero ResNet50 e VGG16, DenseNet121 e InceptionV3.

Per le reti è stata definita una funzione “model_features_extraction” che itera su tutte le immagini, e per ognuna estrae le features. Ciò che si ottiene in output è un *dataframe* in cui ogni riga rappresenta uno user e ogni colonna rappresenta la feature estratta. Per i vari modelli, inoltre, è stato ignorato l’ultimo *layer* relativo alla classificazione. Infine, i file contenenti le features estratte sono stati salvati.

2.2.1 FUNZIONAMENTO DI ResNet50, VGG16, DenseNet121, InceptionV3

ResNet50: è composto da una serie di blocchi residuali. Ogni blocco contiene una combinazione di convoluzioni, pooling e connessioni residuali. Le feature estratte da questi blocchi vengono poi combinate per formare la rappresentazione finale dell'immagine.

VGG16: è composto da una serie di blocchi di convoluzione, ciascuno contenente più livelli di convoluzione 3x3. Questi blocchi sono seguiti da pooling massimo per ridurre la dimensione delle feature map.

L'architettura termina con uno o più livelli fully connected per la classificazione.

DenseNet121: è una particolare tipologia di rete neurale convoluzionale che si distingue per la sua architettura innovativa. A differenza di altre CNN più tradizionali, dove ogni livello è connesso solo al livello successivo, in DenseNet ogni livello è connesso a tutti i livelli successivi. Questa connettività densa tra i livelli conferisce a DenseNet una serie di vantaggi, rendendola particolarmente efficace per compiti di classificazione delle immagini.

InceptionV3: l'idea fondamentale è quella di lasciare che la rete stessa decida quali caratteristiche sono più importanti da estrarre in ogni punto dell'immagine. Questo viene realizzato attraverso i cosiddetti "moduli Inception", che sono come dei "mini-network" all'interno della rete principale.

Ogni modulo Inception contiene diverse convoluzioni con kernel di dimensioni diverse (1x1, 3x3, 5x5) e un layer di pooling. In questo modo, il modulo può estrarre caratteristiche sia locali (con kernel piccoli) che globali (con kernel più grandi).

2.3 ESTRAZIONE DELLE FEATURES CON OpenCV

Per l'estrazione delle features è stata anche utilizzata una libreria software open source per la visione artificiale e l'elaborazione delle immagini. Viene definita una funzione "add_opencv_features" che itera su tutte le immagini, e per ognuna estrae le features. Ciò che si ottiene in output è un *dataframe* in cui ogni riga rappresenta uno user e ogni colonna rappresenta la feature estratta.

2.3.1 FUNZIONAMENTO DI OpenCV

OpenCV tratta le immagini come matrici di numeri, dove ogni numero rappresenta l'intensità di un pixel. Queste matrici possono quindi essere manipolate e analizzate utilizzando una vasta gamma di algoritmi.

2.4 COMBINAZIONE DELLE FEATURES

All'interno del progetto vengono estratte le features e combinate in un unico dataframe, nel primo caso con ResNet50, VGG16 e OpenCV, nel secondo caso con ResNet50, VGG16, OpenCV e DenseNet121 e nel terzo caso con ResNet50, VGG16, OpenCV, DenseNet121 e InceptionV3.

2.5 APPLICAZIONE DELLA DATA AUGMENTATION AL DATASET ORIGINALE

In questa sezione, dato che il dataset era limitato perché costituito solamente da 120 immagini, è stata applicata la *data augmentation*, con l'obiettivo di capire se con questa tecnica fosse possibile ottenere delle prestazioni migliori.

Sono stati applicati:

- Cropping
- Rumore gaussiano

2.6 ESTRAZIONE DELLE FEATURES CON DATA AUGMENTATION

In questa sezione sono stati eseguiti gli stessi passaggi effettuati per l'estrazione delle features senza data augmentation.

Ai file contenenti l'estrazione delle features è stato necessario aggiungere una colonna contenente le labels. Quindi è stato costruito un *dataframe* "df_diag" con una colonna "diag" contenente le rispettive labels. Ogni file relativo alla *features extraction* è costituito da 720 righe. Per come è stato costruito il file, gruppi di 6 righe corrispondono allo stesso user. Quindi, **"df_diag"** è stato costruito in modo che le prime 6 righe sono relative allo user00006, le seconde 6 righe sono relative allo user00007 e così via.

3. CLASSIFICAZIONE

Esclusa la sezione di Import, il file "*Classification"* è suddiviso nelle seguenti parti:

1. Classificazione con modelli di ML (senza data augmentation)
2. Classificazione con modelli di ML (con data augmentation)
3. Concatenazione delle features ottenute senza data augmentation e con data augmentation
4. Classificazione con un modello preaddestrato di InceptionV3
5. Classificazione con rete neurale preaddestrata sul dataset IAM
6. Text Detection

3.1 CLASSIFICAZIONE CON MODELLI DI ML CON E SENZA DATA AUGMENTATION

Sono stati utilizzati i seguenti classificatori:

- Logistic Regression con l1 e l2 penalty
- Decision Tree
- Random Forest
- SVM

Per ogni classificatore è stata calcolata l'*accuracy* assegnando a "random state" 100 valori randomici compresi tra 1 e 100, con l'obiettivo di calcolare un'*accuracy* media.

Inoltre, nel caso specifico di:

- Logistic Regression: è stato fatto variare anche l'iperparametro C
- Random Forest: è stato fatto variare anche l'iperparametro n_estimators
- SVM: è stato fatto variare anche il tipo di kernel

3.1.1 TABELLE CON I RISULTATI

Classificazione con modelli di ML (senza Data Augmentation).

Classificazione con features estratte tramite ResNet50	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Logistic Regression l1	C = 1 C = 100 C = 0,01	0,90 1,00 0,53	0,79 0,78 0,51
Logistic Regression l2	C = 1 C = 100 C = 0,01	0,97 1,00 0,83	0,83 0,79 0,77
Decision Tree	None	1,00	0,70
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,82 0,82 0,81
SVM	linear poly rbf kernel	1,00 0,78 0,75 0,55	0,77 0,75 0,74 0,50

Classificazione con features estratte tramite VGG16	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Logistic Regression l1	C = 1,00 C = 100 C = 0,01	1,00 1,00 0,52	0,98 0,79 0,53
Logistic Regression l2	C = 1 C = 100 C = 0,01	1,00 1,00 0,88	0,77 0,77 0,78
Decision Tree	None	1,00	0,65
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,69 0,69 0,67
SVM	linear poly rbf kernel	1,00 0,83 0,79 0,64	0,77 0,76 0,73 0,63

3.2 CONFRONTO DELL'ACCURACY MEDIA

Viene confrontata l'accuracy media sia sul training set che sul test set considerando dieci valori diversi di random state (ottenuti in maniera randomica) con "Random Forest". L'accuracy media viene calcolata per i tre valori diversi dell'iperparametro n_estimators. La classificazione viene fatta dando in input features estratte con ResNet50, VGG16 (vedere anche il paragrafo precedente), OpenCV, DenseNet121, InceptionV3 e features combinate insieme al modello di classificazione Random Forest, con e senza Data Augmentation.

SENZA DATA AUGMENTATION

Classificazione con features estratte tramite OpenCV	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**

Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,69 0,69 0,68

Classificazione con features estratte tramite DenseNet121	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,79 0,80 0,79

Classificazione con features estratte tramite InceptionV3	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,78 0,78 0,79

Classificazione con features estratte tramite Features combinate	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	1,00 1,00 1,00

CON DATA AUGMENTATION

Classificazione con features estratte tramite ResNet50	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**

Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,78 0,77 0,77

Classificazione con features estratte tramite VGG16	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,74 0,74 0,72

Classificazione con features estratte tramite OpenCV	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,68 0,66 0,67

Classificazione con features estratte tramite DenseNet121	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,74 0,74 0,74

Classificazione con features estratte tramite InceptionV3	**Tuning iperparametri**	**Training accuracy**	**Test accuracy**

Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,77 0,77 0,75

Considerando i dati esposti, si nota come nella classificazione fatta considerando tutte le possibili combinazioni delle features, senza Data Augmentation, l'accuracy sia massima per il training e per il Test definendo un classificatore migliore rispetto a quelli che consideravano features estratte solo con ResNet50 e VGG16.

In particolare, l'accuracy di training indica quanto bene il modello è in grado di prevedere correttamente le etichette delle istanze sul dataset di training, ovvero sui dati che ha utilizzato per apprendere; un'accuracy di training alta ci dice che il modello ha "memorizzato" bene i dati di training.

L'accuracy di test, invece, indica quanto bene il modello è in grado di prevedere correttamente le etichette delle istanze su un dataset di test, ovvero su dati che il modello non ha mai visto durante l'addestramento; un'alta accuracy di test indica che il modello è in grado di generalizzare bene e di fare previsioni accurate su dati non utilizzati durante l'addestramento.

3.3 CONCATENAZIONE DELLE FEATURES OTTENUTE CON E SENZA DATA AUGMENTATION

Sono stati utilizzati i classificatori dandogli in input le features ottenute sia senza *data augmentation* che con *data augmentation*. Questo procedimento è stato applicato solo a ResNet50, che è stata la rete che ha ottenuto le prestazioni migliori.

3.4 CLASSIFICAZIONE CON UN MODELLO PREADDESTRATO DI InceptionV3

Per effettuare la classificazione, in primo luogo, è stata utilizzata la rete neurale InceptionV3, già addestrata sul dataset ImageNet. Dopo aver eliminato il layer di output, sono stati aggiunti:

- Un global average pooling layer
- Un layer dropout (con r = 0,5)
- Un layer denso, formato da 256 nodi
- Un altro layer di dropout (con r = 0,5)
- Un layer di output, formato da 1 nodo per la classificazione binaria e con funzione di attivazione sigmoide

Tutti i layer della rete InceptionV3 sono stati congelati, e solo i layer aggiunti sono stati addestrati. La rete è stata addestrata su 200 epoche.

Dopodiché, per aumentare l'accuracy, i pesi di InceptionV3 sono stati scongelati in modo tale da poter riaddestrare l'intera rete, utilizzando sempre 200 epoche.

3.5 CLASSIFICAZIONE CON UNA RETE NEURALE PRE-ADDESTRATA SUL DATASET I-AM

Per effettuare la classificazione è stata utilizzata in seconda istanza una rete neurale pre-addestrata su un dataset composto da forme di testo scritte a mano in inglese.

Dopo aver eliminato il layer di output della rete caricata, è stato aggiunto:

- Un layer denso
- Un layer di output con funzione di attivazione sigmoide

Inizialmente, i pesi del modello caricato sono stati congelati e sono stati addestrati sul dataset relativo alla disgrafia solo i layer aggiunti.

La rete è stata addestrata su 200 epoche, e per ogni epoca è stata stampata l'accuracy e la loss.

Successivamente, sono stati scongelati anche i pesi del modello caricato che è stato addestrato nuovamente su 200 epoche. In questo modo i pesi sono stati aggiornati partendo da un'inizializzazione non randomica, ma dall'inizializzazione relativa all'addestramento sul dataset IAM.

3.6 TEXT DETECTION

Per eseguire il task della text detection è stato importato un tool per il riconoscimento ottico dei caratteri chiamato "pytesseract". Attraverso la funzione "detect_text_sentences" è stato possibile associare ad ogni immagine una lista di dizionari, uno per ciascuna parola o carattere riconosciuto. Ogni dizionario contiene informazioni dettagliate sulla disposizione del testo riconosciuto nell'immagine, come ad esempio le coordinate dei bounding box delle parole, le coordinate dei caratteri individuati, il testo estratto e altre informazioni utili. Dopodiché, la funzione "save_text_sentences_as_images" permette di caricare l'immagine e i dati di rilevamento del testo ottenuti tramite la funzione "detect_text_sentences". Essa estrae ciascuna parola dall'immagine originale utilizzando le coordinate del bounding box e salva ciascuna parola come un'immagine separata nelle directory di output.

Successivamente, la funzione "save_image" richiama entrambe le funzioni precedentemente descritte per ogni immagine del dataset.

La funzione "process_image" ha permesso di ingrandire le immagini contenenti le parole estratte e di ridimensionarle aggiungendo il padding in modo da avere tutte la stessa dimensione, adeguata alla rete.

Infine, è stata riutilizzata la rete InceptionV3 per effettuare la classificazione. Dopo aver eliminato il layer di output, sono stati aggiunti:

- Un layer denso, formato da 256 nodi

- Un layer di dropout (con r = 0,5)

- Un layer di output, formato da 1 nodo per la classificazione binaria e con funzione di attivazione sigmoide

Tutti i layer della rete InceptionV3 sono stati congelati, e solo i layer aggiunti sono stati addestrati. La rete è stata addestrata su 200 epoche.

Dopodiché, per aumentare l'accuracy, i pesi di InceptionV3 sono stati scongelati in modo tale da poter riaddestrare l'intera rete, utilizzando sempre 200 epoche.

4 ACCURACY OTTENUTA DALLE TRE RETI ADDESTRATE

Nel seguito sono mostrati i risultati ottenuti da InceptionV3, dalla rete preaddestrata con il dataset IAM e dalla Text Detection:

	Training accuracy	Validation accuracy
InceptionV3	1,0	0,83
Rete preaddestrata con IAM	0,99	0,9
Text Detection	0,99	0,62

5 SVILUPPO DEL PROGETTO DI SISTEMI INTELLIGENTI

Per poter fare delle indagini accurate in merito alla possibile disgrafia di alcuni soggetti volontari, in numero pari a 14, abbiamo proceduto a far scrivere loro una frase su un foglio (Che bella giornata). Per poter associare ciascuna frase al soggetto che l'ha scritta si è proceduto a realizzare un file 'Identificativi.txt' all'interno del quale è stato riportato l'ID del candidato e il suo nome con il cognome puntato per motivi di privacy. Per poter analizzare le varie scritture abbiamo proceduto a fare delle foto, inserite poi all'interno della cartella **test_images**; a ciascuna foto in formato .jpg è stato dato un nome corrispondente all'ID del soggetto testato (numero identificativo). Conseguentemente abbiamo caricato la cartella sul Drive affinché il sistema potesse afferire corettamente alle immagini.

5.1 UTILIZZO DEI FILE

Per poter realizzare il lavoro ci siamo serviti dei file Classification.ipynb e Features_extraction.ipynb, precedentemente spiegati, presenti all'interno di un drive. Si tratta di due file di origine Jupyter sviluppati in Colab che è un servizio di notebook Jupyter ospitato che non richiede alcuna configurazione e offre accesso senza costi alle risorse di computing, incluse GPU e TPU.

5.1.1 USO DEL FILE CLASSIFICATION

Attraverso il file **Classification** siamo stati in grado di salvare il modello di classificazione Random Forest in locale per consentire la riutilizzabilità e l'efficienza operativa nelle fasi successive del progetto. Una volta addestrato il modello sui dati di training, è stato serializzato e salvato utilizzando joblib.

Questo approccio presenta diversi vantaggi:

- **Evitare il ritraining**: non è necessario riaddestrare il modello ogni volta che si vuole effettuare una predizione, risparmiando tempo e risorse computazionali.
- **Portabilità**: il file .pkl può essere facilmente caricato su altre macchine o ambienti, garantendo la replicabilità del sistema.
- **Integrazione**: permette una più semplice integrazione con altri moduli del progetto (ad esempio, classificazione batch di immagini) senza dipendere dal notebook o dallo script di training.
- **Automazione**: è fondamentale in pipeline automatizzate dove il modello deve essere caricato e utilizzato in produzione o durante test su nuovi dati.

Il modello Random Forest, una volta salvato in locale, è stato caricato sul drive per poter essere utilizzato correttamente all'interno del file **Features_extraction.**

5.1.2 USO DEL FILE FEATURES_EXTRACTION

Attraverso l'utilizzo del file Features_extraction è stato possibile fare la classificazione delle immagini manoscritte; è stata implementata una pipeline che ha previsto l'utilizzo combinato di modelli preaddestrati e tecniche di visione artificiale tradizionale. In particolare, sono stati caricati i modelli **ResNet50**, **VGG16**, **DenseNet121** e **InceptionV3**, tutti preaddestrati su ImageNet, un'ampia base di dati con più di 14 milioni di immagini, al fine di estrarre **feature semantiche di alto livello** da ciascuna immagine. A queste si sono aggiunte **feature derivate da OpenCV**, quali istogrammi di intensità e numero di contorni rilevati, per completare la rappresentazione visiva con caratteristiche di tipo strutturale e locale.

Tutte queste informazioni eterogenee sono state successivamente unificate in un unico vettore di feature, concatenando le rappresentazioni estratte dai diversi modelli. Questa operazione ha permesso di combinare i punti di forza delle architetture deep learning con elementi di visione classica, offrendo una descrizione più ricca e informativa delle immagini da classificare.

Una volta ottenuti i vettori completi, è stato utilizzato il modello di classificazione Random Forest precedentemente addestrato, salvato in

locale e inserito poi nel Drive per assegnare a ciascuna immagine una classe: **disgrafico** o **non disgrafico**.

Per garantire una valutazione sistematica e facilmente consultabile dei risultati, il sistema salva automaticamente:

- un **file Excel** contenente il nome dell'immagine, corrispondente all'ID dei soggetti testati, e la rispettiva predizione;
- un **file testuale** che contiene un riepilogo comprensibile con le percentuali di ciascuna classe e il numero totale di immagini valide elaborate;
- un **grafico a barre** che mostra il numero assoluto di predizioni per ciascuna classe;
- un **grafico a torta** che rappresenta le stesse informazioni in forma percentuale.

5.2 RISULTATI OTTENUTI

Come riportato in 5.1.2 i risultati ottenuti dall'analisi sono contenuti all'interno di 4 file diversi; questi file riportano però un unico risultato, cioè la non presenza di soggetti disgrafici tra coloro che si sono sottoposti al test.

All'interno del file excel **prediction.xlsx** è possibile notare come singolarmente ciascun soggetto sia stato considerato non disgrafico; in una colonna viene infatti riportato l'identificativo dell'immagine associata a una frase scritta dal soggetto sottoposto al test e in un'altra colonna viene riportata la predizione, in termini di classificazione, fatta relativamente a ciascuna immagine (disgrafico, non_disgrafico).

Nel file testuale **summary_result.txt** viene riportato il calcolo delle percentuali di distribuzione delle due classi previste: *disgrafico* e *non disgrafico*. Queste percentuali vengono ottenute rapportando il numero di immagini appartenenti a ciascuna classe al totale delle immagini correttamente analizzate. Il file generato include:

- Un'intestazione esplicita *"Riepilogo Classificazioni"*,
- Le percentuali relative a ciascuna classe predetta,

- Le percentuali specifiche dei *disgrafici* e dei *non disgrafici* evidenziate separatamente per chiarezza,
- Il numero totale di immagini valide elaborate, ovvero quelle per cui è stato possibile generare una predizione senza errori tecnici.

Nel file di immagine **grafico_barre_predizioni.png**, dato che tutte le 14 immagini sono state classificate come "non_disgrafico", viene presentato un grafico a barre che ha:

- Una sola barra, etichettata come non_disgrafico
- Altezza pari a 14 (il numero di immagini classificate in quella classe)
- Nessuna barra per la classe disgrafico, poiché nessuna immagine è stata classificata in questo gruppo

Lo scopo di questo grafico è pertanto offrire una rappresentazione immediata e facilmente interpretabile della distribuzione delle classi predette dal modello. È uno strumento utile per evidenziare eventuali sbilanciamenti tra le classi e per valutare il comportamento del classificatore in fase di test.

Nel file di immagine **grafico_torta_predizioni.png**, vi è un grafico a torta che ha:

- una sola fetta, etichettata "non_disgrafico – 100%"
- Non comparirà alcuna fetta per **"disgrafico"**, poiché nessuna immagine è stata classificata in quella categoria
- Questo evidenzia visivamente che tutte le immagini testate sono state considerate non disgrafici dal modello

Il suo scopo principale è visualizzare in maniera intuitiva e immediata quanto il modello abbia assegnato a ciascuna classe all'interno del dataset di test. Mentre il grafico a barre evidenzia **i conteggi assoluti**, il grafico a torta mostra **le proporzioni relative (%)**, rendendo chiaro quanto pesa ciascuna classe rispetto al totale.