UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria

Corso di Laurea in Ingegneria Informatica

PROGETTO MACHINE LEARNING E DEEP LEARNING

RICONOSCIMENTO DELLA DISGRAFIA

Alessandra Marchi 523478

Alessio Pallotta 522514

Anno Accademico 2022/2023

1. PROBLEM STATEMENT

A partire da un dataset di immagini con caratteri scritti a mano, l'obiettivo del progetto è quello di classificare i pazienti in disgrafici o non disgrafici.

Il progetto è stato organizzato in due file ipynb:

- Il file "***Features_extraction"***, in cui viene effettuata l'estrazione delle features.
- Il file "***Classification"***, in cui viene fatta la classificazione.

2. ESTRAZIONE DELLE FEATURES

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

1. Costruzione dataset
2. Estrazione features con CNN
3. Applicazione della *data augmentation* al dataset originale
4. Estrazione features con *data augmentation*

Nel seguito viene illustrata e spiegata ogni parte.

2.1 COSTRUZIONE DATASET

In questa sezione viene caricato il file "users" che contiene i dati dei pazienti. Il file è organizzato nel seguente modo: ogni riga rappresenta 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

SALVATAGGIO DI TUTTE LE IMMAGINI CONVERTITE

È stata 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)

È stata definita una funzione “view_image” che ci permette di costruire l’immagine a partire dalle coordinate contenute nella sessione estratta grazie alla funzione precedente. Dopodiché, tramite la funzione “save_image”, queste immagini sono state salvate nel drive.

2.2 ESTRAZIONE FEATURES CON CNN

Per l’estrazione delle features sono state utilizzate due architetture deep già implementate: ResNet50 e VGG16.

Per entrambe 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 entrambi i modelli è stato ignorato l’ultimo *layer* relativo alla classificazione. Infine, sempre per entrambi i modelli i rispettivi file contenenti le features estratte sono stati salvati.

2.3 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

2.4 ESTRAZIONE 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

Nel seguito viene illustrata e spiegata ogni parte

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* utilizzando 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	0,81 0,80 0,80
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 C = 100 C = 0,01	0,94 1,00 0,53	0,71 0,72 0,52
Logistic Regression l2	C = 1 C = 100 C = 0,01	1,00 1,00 0,85	0,76 0,75 0,72
Decision Tree	None	1,00	0,61
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,67 0,67 0,66
SVM	linear poly rbf kernel	1,00 0,80 0,75 0,54	0,75 0,68 0,62 0,49

CLASSIFICAZIONE CON MODELLI DI ML (CON DATA AUGMENTATION)

Classificazione con features estratte tramite ResNet50	Tuning iperparametri	Training accuracy	Test accuracy
Logistic Regression l1	C = 1 C = 100 C = 0,01	1,00 1,00 0,52	0,94 0,81 0,53
Logistic Regression l2	C = 1 C = 100 C = 0,01	0,99 1,00 0,84	0,82 0,80 0,80
Decision Tree	None	1,00	0,64
Random Forest	n_estimators = 100 n_estimators = 80 n_estimators = 50	1,00 1,00 1,00	0,78 0,78 0,78
SVM	linear poly rbf kernel	1,00 0,79 0,77 0,72	0,79 0,75 0,74 0,71

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	0,74 0,74 0,72
SVM	linear poly rbf kernel	1,00 0,83 0,79 0,64	0,77 0,76 0,73 0,63

3.2 CONCATENAZIONE DELLE FEATURES OTTENUTE SENZA DATA AUGMENTATION E CON 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.

3.3 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.

3.4 CLASSIFICAZIONE CON 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 70 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 40 epoche. In questo modo i pesi sono stati aggiornati partendo da un'inizializzazione non randomica, ma dall'inizializzazione relativa all'addestramento sul dataset IAM.

3.5 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", 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 da EfficientNetB0

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