Image Segmentation

Lorenzo Sciarretta

June 5, 2023

Abstract:

In questo progetto si presenta un tentativo di segmentazione dei reni e tumori nelle immagini mediche utilizzando l'architettura Fully Convolutional Network (FCN) basata su ResNet101. Il processo di segmentazione è fondamentale per la diagnosi e il monitoraggio di patologie renali, consentendo una valutazione precisa. Il codice è strutturto come segue: 'preprocessing' delle immagini e delle relative segmentazioni, creazione di un dataset personalizzato ed infine 'training' del modello .

1 Preprocessing

- Per prima cosa importo le librerie necessarie, inclusi i moduli di PyTorch e Nibabel per la manipolazione delle immagini.
- Definisco la funzione `make_img_path(cid)` che restituisce il percorso del file di imaging corrispondente a un determinato caso.
- Definisco la funzione `make_seg_path(cid)` che restituisce il percorso del file di segmentazione corrispondente a un determinato caso.
- Creo i percorsi delle cartelle di output per i dati preelaborati: train e valid.
- Pre-elaborare le immagini di addestramento , salvandole come file numpy (.npy) nelle rispettive cartelle.

2 Dataset

- Definisco la classe `KitsDataset` che rappresenta il dataset personalizzato per il progetto.
- Fornisco i percorsi delle cartelle contenenti le immagini e le segmentazioni.
- utilizzo il metodo `__len__` per ottenere la lunghezza del dataset.

- definisco la funzione `__getitem__` per ottenere un campione dal dataset, con le caratteristicche necessarie per la mia rete.
- Utilizzo `torch.tensor` per caricare le immagini e i segmenti come tensori PyTorch.
- Applico trasformazioni e normalizzazioni all'immagine.
- Restituisco infine un campione `sample` contenente l'immagine e la maschera.

3 Training del modello

- Definisco la funzione `set_parameter_requires_grad` per impostare il flag `requires_grad` dei parametri del modello.
- Definisco la funzione `createModel` per creare il modello di segmentazione basato sull'architettura FCN-ResNet101.
- Definisco la funzione `collate_fn` per gestire il raggruppamento dei campioni all'interno di un batch e per gestire eventuali errori.
- Definisco la funzione `train_model` per addestrare il modello.
- Itero su ogni epoca di addestramento e validazione(Durante ogni epoca, calcolae perdite e le metriche di valutazione).
- Salva i pesi del modello migliore basato sulla perdita di validazione più bassa.
- Alla fine del training, caricare il modello con i pesi migliori.

4 Main

- Creo effetivamente il dataset di addestramento e di validazione utilizzando la classe `KitsDataset`.
- Definisco i data loader per l'addestramento e la validazione utilizzando i dataset creati.
- Creo il modello di segmentazione utilizzando la funzione `createModel`.
- Definisco l'ottimizzatore e la funzione di perdita per l'addestramento del modello.
- Addestro il modello utilizzando la funzione `train_model`.
- Infine effettuo la segmentazione delle nuove immagini utilizzando il modello addestrato.

Riferimenti

Preprocessing:

- Lezioni 1-6 del corso e implementazioni di funzioni trovate online.

Rete neurale:

- Lezione 7 del corso.