PROGETTO CCAI

Luca Dimarco
Dario Lomonaco
Antonino Salemi

PROBLEMA

Creazione di un modello per la classificazione d'immagini.

SFIDA PRINCIPALE

Domain Shift - Differenze significative tra i dati di addestramento e quelli di test.

COMPOSIZIONE DATASET

TRAINING: 1600 immagini distribuite in 8 categorie diverse.

TEST: 800 immagini con lo stesso sfondo delle immagini di addestramento.

RISULTATO DESIDERATO

Sviluppare un modello che possa adattarsi efficacemente alle differenze tra i due domini, consentendo una classificazione accurata delle immagini di test.

APPROCCI UTILIZZATI

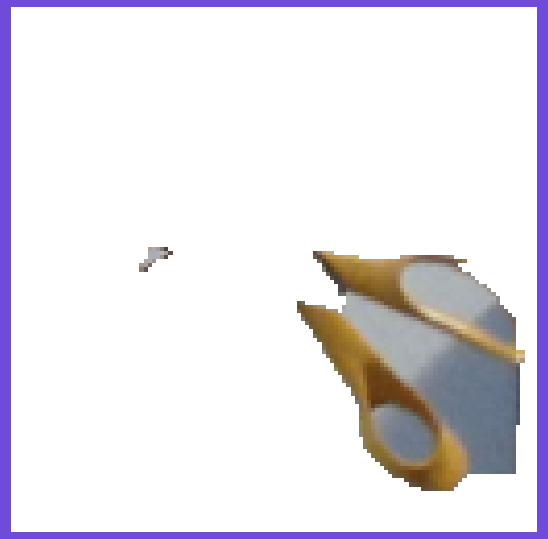
- BACKGROUND REMOVING
- IMAGE VARIATION 1.0 (APPROCCIO GREZZO)
- IMAGE VARIATION 2.0 (APPROCCIO MIRATO)

BACKGROUND REMOVING

Questo approccio consiste nel trattare lo sfondo come rumore e quindi rimuoverlo.

Nel Dataset avremo 1600 immagini con sfondo e 1400 senza sfondo.

Questo perché la rimozione dello sfondo toglie dettagli all'immagine come se applicassimo delle trasformazioni, quindi abbiamo bisogno di compensare con le immagini originali tale mancanza di informazioni(Questo solo per le immagini che hanno sfondo diverso da quello di test).



BACKGROUND REMOVING

Altre tecniche utilizzate:

- **Transform;**
    - **RandomHorizontalFlip;**
- **Fine tuning.**

Link al modello

FUNZIONE PRINCIPALE

Questa funzione prende in input un percorso relativo ad un'immagine (image_path), una bounding box (bbox) che specifica la regione di interesse dell'immagine, e un valore booleano (noback). Se noback è impostato su 0, la funzione esegue la segmentazione dell'immagine per rimuovere lo sfondo intorno alla bounding box utilizzando l'algoritmo GrabCut. Successivamente, l'immagine risultante viene ritagliata in base ai valori della bounding box.

```
def crop_image(image_path, bbox, noback):
    image = cv2.imread(image_path)
    x_min, y_min, x_max, y_max = bbox
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    if noback==0:
        mask = np.zeros(image_rgb.shape[:2], np.uint8)
        bgd_model = np.zeros((1, 65), np.float64)
        fgd_model = np.zeros((1, 65), np.float64)
        rectangle = (x_min, y_min, x_max - x_min, y_max - y_min)
        cv2.grabCut(image_rgb, mask, rectangle, bgd_model, fgd_model,1, cv2.GC_INIT_WITH_RECT)
        mask_fg = np.where((mask == 3) | (mask == 1), 255, 0).astype('uint8')
        image_rgb = cv2.bitwise_and(image_rgb, image_rgb, mask=mask_fg)
    cropped_image = image_rgb[y_min:y_max, x_min:x_max]
    res = np.array(cropped_image) / 255.0
    return res
```

IMAGE VARIATIONS 1.0

Questo approccio consiste nel modificare il tono, la luminosità e il contrasto dell'immagine in un range casuale in modo da avere molte variazioni per ogni elemento del dataset.

La nostra idea era quella di far vedere al modello immagini di vario tipo in modo che potesse generalizzare meglio sul dominio di test.

Altre tecniche utilizzate:

- **Fine tuning;**
- **Resize and Crop;**

Link al modello

TECNICA PRINCIPALE

```
transf = T.ColorJitter(brightness=(0,1),contrast=(0,1),hue=(-0.5,.5))
```

Controllare se è vera, l'ho scritta al momento

IMAGE VARIATIONS 2.0

Il modello precedente aveva un difetto particolare, dipendeva troppo dalla randomicità dei dati di train, questa faceva oscillare le performance dal 90 al 99 percento, quindi dopo una ricerca manuale di parametri ottimali abbiamo sviluppato un modello con una media piu alta.

Altre tecniche utilizzate:

- **Resize and Crop**
- **Fine Tuning**

Link al modello

TECNICA PRINCIPALE

```
transf = T.ColorJitter(brightness=(0.1),contrast=(0.9), saturation=(1.3),hue=(-0.5,.5))
```

RISULTATI

STEP	BR	IV
1	99.25	99.125
2	99.625	99.75
3	99.75	--
4(solo fc)	99.875	99.875

TASSO ERRORE

1 /800