B Používateľská príručka

B.1 Úvod

Praktická časť tejto práce je implementovaná v jazyku Python s využitím knižníc TensorFlow, matplotlib, numpy, pandas, Pillow.

B.2 Predpoklady

Na spustenie tohto projektu je potrebné mať nainštalovaný Python a Git.

- Python 3.11 (64-bit)
- Git (na klonovanie repozitára)

B.3 Príprava projektu

Pred samotným spustením projektu je nutné vykonať 3 kroky a to získať zdrojový kód projektu, získať dataset a prípadne aj už natrénované modely. Nakoľko všetky tieto komponenty nebolo kvôli ich veľkosti možné vložiť ako prílohu, tak tu opisujeme postup, ako získať jednotlivé prístupy.

B.3.1 Získanie zdrojového kódu projektu

Ak si chce používateľ vyskúšať naše softvérové riešenie tohto projektu, prvým krokom je získanie samotného zdrojového kódu. To sa dá dvomi spôsobmi:

- Pokiaľ používateľ disponuje aj prílohou tohto projektu, tak si len jednoducho stačí súbor retinal-segmentation.zip rozbaliť do ľubovoľného priečinku, z ktorého sa bude následne projekt spúšťať.
- Druhou možnosťou je si projekt so zdrojovým kódom stiahnuť z Google Disk na *tomto odkaze* a rozbaliť ho do ľubovoľného priečinku s použitím hesla `Qa4%!ae*&Ke3ug` .

Po získaní zdrojového kódu bez ohľadu na spôsob jeho získania, je pre správne fungovanie projektu vhodné vytvoriť a aktivovať virtuálne prostredie, aby sme zabezpečili, že neskôr nainštalované knižnice budú mať správne verzie, vyhovujúce účelom tohto projektu.

```
python -m venv venv
source venv/bin/activate # Na Windowse: venv\Scripts\activate
```

Po aktivácii virtuálneho prostredia je potrebné nainštalovať potrebné Python knižnice, ktoré sú spoločne s požadovanými verziami uložené v requirements.txt súbore.

```
pip install -r requirements.txt
```

B.3.2 Získanie datasetov

V prípade, že si používateľ chce natrénovať nový model, tak je k tomu potrebné mať aj príslušné datasety. Vzhľadom na ich veľkosť však nebolo možné ich priložiť ako prílohu, preto sú na stiahnutie dostupné na *tomto odkaze*.
Pre rozbalenie súboru, ktorý je vo formáte .zip je potrebné použiť heslo `rEWN!f72B!4n#&` . Priečinok po rozbalení obsahuje všetkých päť datasetov, ktoré sú opísané v tejto práci a aj dataset 'macula', ktorý obsahuje 5 snímkov čistých makúl aj s maskami. Používateľovi stačí vybraný dataset skopírovať priamo do koreňového adresára zdrojového kódu projektu (retinal-segmentation). Každý z piatich priečinkov s datasetom obsahuje tri podpriečinky pre trénovacie (train), validačné (eval) a testovacie (test) dáta. V zdrojom kóde bude neskôr potrebné upraviť cesty práve k týmto priečinkom.

B.3.3 Získanie natrénovaných modelov

V prípade, že má používateľ záujem o prácu s niektorým z natrénovaných modelov, ktoré sme vytvorili v rámci tejto práce, tak sú dostupné na stiahnutie na *tomto odkaze*.
Pre rozbalenie súboru, ktorý je vo formáte .zip je potrebné použiť heslo `Tfn9ws5fU3DfGk` . Upozorňujeme, že aj v tomto prípade je potrebné disponovať aj samotným datasetom, nestačí iba zdrojový kód a model. Priečinok po rozbalení obsahuje všetky modely U-Net siete, ktoré sú opísané v tejto práci.
Každý z modelov je súbor s príponou .h5, čiže Keras model. Vybraný model stačí iba vložiť do koreňového adresára zdrojového kódu projektu (retinal-segmentation), prípadne si tu používateľ môže pre prehľadnosť vytvoriť nový podpriečinok (napr. 'model') a samotný model vložiť tam. V ďalšej časti tejto príručky je postup, ako v zdrojovom kóde správne nastaviť cestu k modelu. Po vykonaní týchto krokov je projekt pripravený na použitie.
Po vykonaní týchto krokov je projekt pripravený na použitie.

B.4 Použitie natrénovaného modelu

Pre použitie natrénovaného modelu je potrebné upraviť a spustiť súbor `predict.py` v adresári `unet`. Do súboru je potrebné doplniť cesty k modelu `MODEL_PATH` na ktorom sa bude predikovať, cestu k datasetu `BASE_DIR` a cestu k vzorovej maske `MASK_PATH` z datasetu, aby bolo možné namapovať farby masky na vrstvy sietnice.

Ak v datasete nie je testovacia množina pomenovaná `test`, je potrebné zmeniť aj konštantu `TEST_DIR` na správnu cestu.

Pre správne fungovanie predikovania je potrebné aby `TEST_DIR` obsahoval dve zložky `img` a `mask` v ktorých bude obrázok a maska pomenovaná rovnako.

V prípade potreby je možné upraviť ostatné konštanty na začiatku súboru, ktoré ovplyvňujú správanie modelu.

Príklad nastavenia konštánt:

```
MODEL_PATH = 'model/checkpoints/model.h5'
MASK_PATH = 'dataset/sanghai_dataset/train/mask/10_R_02_left.bmp'
BASE_DIR = 'dataset/sanghai_dataset'
TEST_DIR = os.path.join(BASE_DIR, 'test')
IMG_HEIGHT, IMG_WIDTH = 256, 256
BATCH_SIZE = 16
NUM_PREDICTIONS = 8
```

Po úprave súboru je možné spustiť predikcie pomocou príkazu:

```
python unet/predict.py
```

Výsledky predikcie budú zobrazené v konzole.

B.5 Trénovanie nového modelu

Pre trénovanie nového modelu je potrebné upraviť a spustiť súbor `train.py` v adresári `unet`.

Do súboru je potrebné doplniť cesty k datasetu, kde sa nachádzajú trénovacie `TRAIN_DIR`, validačné `EVAL_DIR` a testovacie dáta `TEST_DIR` (ak sú iné ako je v súbore preddefinované). Rovnako je potrebné doplniť cestu `MODEL_PATH`, kde sa bude model ukladať počas trénovania a cestu `MASK_PATH` k vzorovej maske z datasetu, aby bolo možné namapovať farby masky na vrstvy sietnice.

V prípade potreby je možné upraviť ostatné konštanty na začiatku súboru, ktoré ovplyvňujú správanie modelu.

Príklad nastavenia konštánt:

```
MODEL_PATH = 'models/model.h5'
MASK_PATH = 'dataset/sanghai_dataset/train/mask/10_R_02_left.bmp'
BASE_DIR = 'dataset/sanghai_dataset'
TEST_DIR = os.path.join(BASE_DIR, 'test')
TRAIN_DIR = os.path.join(BASE_DIR, 'test')
EVAL_DIR = os.path.join(BASE_DIR, 'eval')

IMG_HEIGHT, IMG_WIDTH = 256, 256
BATCH_SIZE = 16
NUM_CLASSES = 10
BUFFER_SIZE = 100

EPOCHS = 20
ES_PATIENCE = 3
```

Po úprave súboru je možné spustiť trénovanie modelu pomocou príkazu:

```
python unet/train.py
```

Výsledný model ako aj priebežné checkpointy trénovania budú uložené v `MODEL_PATH`.

B.6 Augmentácia

Pre efektívnu augmentáciu celého datasetu je vytvorený modul `unet/utils/augmentation.py`, ktorý poskytuje funkcie pre rôzne typy augmentácie obrázkov. Rovnako poskytuje možnosť vytvorenia nového datasetu s augmentovanými obrázkami a maskami z pôvodného datasetu pomocou funkcie `generate_augmented_dataset`. Táto funkcia vytvorí nový dataset v zadanom adresári, ktorý bude obsahovať obrázky a masky po aplikácií vybraných metód augumentácie.
Príklad použitia:

```
augmentation_lists = [
    [change_brightness, flip],
    [change_brightness],
    [change_contrast, scale]
]
```

```
# Set the output directory for the augmented dataset
output_dir = '../augmented_dataset'

# Directory containing the original images
img_dir = '../original_images'

# Directory containing the original masks
mask_dir = '../original_masks'

# Call the generate_augmented_dataset function
generate_augmented_dataset(augmentation_lists, output_dir, img_dir, mask_dir)
```

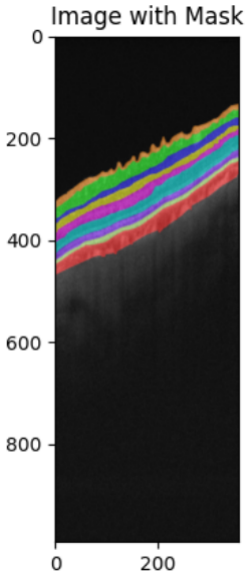
Vysvetlenie: - `augmentation_lists` - zoznam zoznamov funkcií, ktoré sa majú aplikovať na obrázky a masky. Každý zoznam obsahuje funkcie, ktoré sa majú aplikovať na obrázok a masku zároveň. - `output_dir` - cesta k výstupnému adresáru, kde sa uložia augmentované obrázky a masky. - `img_dir` - cesta k adresáru s pôvodnými obrázkami. - `mask_dir` - cesta k adresáru s pôvodnými maskami.

B.7 Vizualizácia výsledkov

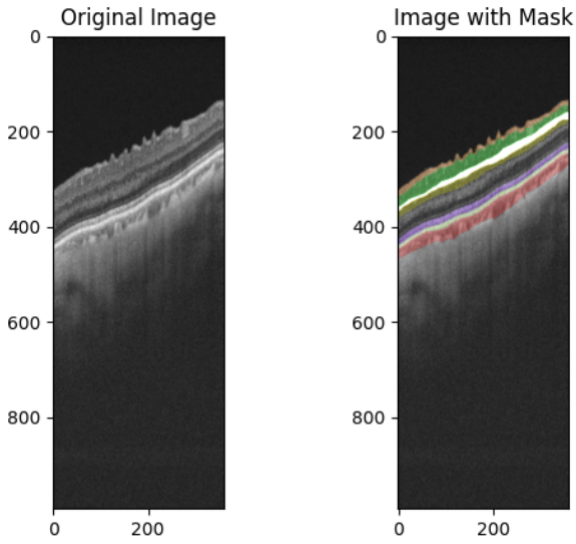
Pre lepšiu vizualizáciu výsledkov je vytvorený modul `unet/utils/visualizer.py`, ktorý poskytuje funkcie na zobrazenie obrázkov a masiek. Modul poskytuje mnohé možnosti zobrazenia.
Príklad použitia:

```
display_image_with_mask(image, mask)


    # or
    CUSTOM_MAPPING = {
        (77, 77, 77): (255, 255, 255),  # Layer 3 (white)
    }
    display_image_with_mask(image,
           mask,
           display_original=True,
           alpha=0.25,
           color_mapping=CUSTOM_MAPPING,
```

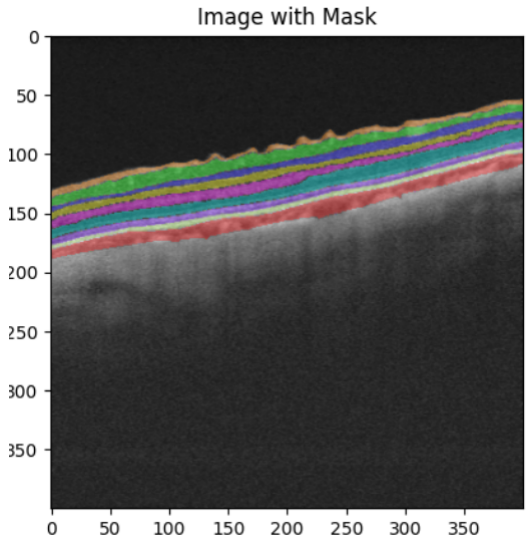
```
        display_layers_mask=[0, 1, 1, 1, 1, 0, 0, 1, 1, 1])

# or
display_image_with_mask(image,
              mask,
              target_size=(400, 400),
              alpha=0.35,
              display_layers_mask=[0, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```



Obr. B.1: Výstup z prvého spustenia vizualizácie



Obr. B.2: Výstup z druhého spustenia vizualizácie



Obr. B.3: Výstup z tretieho spustenia vizualizácie

Vysvetlenie `display_image_with_mask()` funkcie:

- `image` - obrázok, ktorý sa má zobraziť.
- `mask` - maska, ktorá sa má zobraziť.
- `mask_to_compare` - maska, ktorá sa má zobraziť vedľa pôvodnej masky na porovnanie.
- `alpha` - priehľadnosť masky.
- `target_size` - veľkosť obrázku a masky.
- `display_original` - zobraziť pôvodný obrázok.
- `color_mapping` - mapovanie farieb pre zobrazenie masky.
- `display_layers_mask` - zobraziť iba vybrané vrstvy masky.
- `titles` - názvy obrázkov a masiek.