UNet 用データセットの整理

今回実験的に作成する UNet に学習させるためのデータセットを用意する.

2018 Data Science Bowl

すべてのデータをダウンロードすると、以下のようなフォルダ構造を持つ.

1. stage1_sample_submission.csv.zip
2. stage1_solution.csv
3. stage1_test
4. stage1_train
5. stage1_train_labels.csv
6. stage2_sample_submission_final.csv
7. stage2_test_final

そのうち，`stage1_train` を解凍してみると、

```
<stage1_train>
|-- 00ae65
|       |-- images
|       |       |-- 00ae65.png
|       |-- masks
|               |-- 0f33d2a.png
|               |-- 0fe691c.png
|               |-- ...
|
|-- 0a7d30
|       |-- images
|       |       |-- 0a7d30.png
|       |-- masks
|               |-- 0adbf56.png
|               |-- 1ed5b9a.png
|               |-- ...
|
|-- 0acd2c
|       |-- images
|       |       |-- 0acd2c.png
|       |-- masks
|               |-- 2f5eb1.png
|               |-- 3fa2a8.png
|               |-- ...
|...
```

のようになっている.

ここから，ディレクトリおよびファイル名を次のように変更する.

このとき，masks ディレクトリ内に存在した画像名がすべて images ディレクトリ内の画像名と同一になっていることに注意．

```
<stage1_train>
|-- images
|       |-- 00ae65.png
|       |-- 0a7d30.png
|       |-- 0acd2c.png
|-- masks
        |-- 0
        |    |-- 00ae65.png
        |    |-- 0a7d30.png
        |    |-- 0acd2c.png
        |    |-- ...
        |
        |-- 1
        |    |-- 00ae65.png
        |    |-- 0a7d30.png
        |    |-- 0acd2c.png
        |    |-- ...
        |
        |-- 2
        |    |-- 00ae65.png
        |    |-- 0a7d30.png
        |    |-- 0acd2c.png
        |    |-- ...
        |-- ...
```

---

zipファイルを自動的に展開する

ダウンロードしてきたデータはすべて zip 形式で保存されいる．これは，オリジナルデータとして保存しておく分には問題ないが，画像を確認する場合にはいちいち展開しなくてはいけないので，手間がかかる．そこで，zip ファイルを自動的に解凍する関数を作成する．

Python, zip関数の使い方: 複数のリストの要素をまとめて取得 を参考に，zipfile ライブラリを用いて以下のような関数を作成した．

```
def unpack(self, file_name: str, create_dir: bool=True):
    """originalディレクトリ内のzipファイルを展開するための関数
    Arg:
        file_name(str): original ディレクトリ内の zip ファイル名，
        create_dir(bool optional): 解凍するときに、解凍前のファイル名と同じ名前のディレクトリを作
成する。default to True.
    """
    zip_path = os.path.join(self.org_path, file_name + '.zip')
    unpack_path = self.parent_path

    if create_dir:
```

```
        try:
            unpack_path = os.path.join(self.parent_path, file_name)
            os.mkdir(unpack_path) # create directry
        except FileExistsError as e:
            print("ディレクトリ作成時にエラーが発生しました: ", e)
            return

    with zipfile.ZipFile(zip_path) as existing_zip:
        existing_zip.extractall(unpack_path)
```

この関数を用いることで，展開前の `zip` ファイルの名前を持つディレクトリ内にファイルを自動的に解凍することができる．

---

images ディレクトリ内の png 画像を別のディレクトリに移動する．

次に，解凍したディレクトリから新しい訓練データセットを作成するために、ファイルを整理していく．そこで，解凍したデータ内のそれぞれのラベル名が付いたディレクトリ内にある `images` ディレクトリ内の画像から移動していく．

1. 解凍したデータ内のディレクトリ一覧を取得する．

Pythonで条件を満たすパスの一覧を再帰的に取得するglobの使い方 を参考に，以下のプログラムによってファイル一覧を取得する．

```
$ > list_dir_path = glob(os.path.join(src, '**'))
```

これによって，以下のようなディレクトリ名の一覧が取得できる．

```
'DataScienceBowl\\stage1_train\\c901794d1a421d52e5734500c0a2a8ca84651fb93b19cec2f4
11855e70cae339'
'DataScienceBowl\\stage1_train\\c96109cbebcf206f20035cbde414e43872074eee8d839ba214
feed9cd36277a1'
'DataScienceBowl\\stage1_train\\c9f305be17312bdb9530fb4f1adc6d29730ddbe0e74730cbf0
31de174bf437b7'
'DataScienceBowl\\stage1_train\\cab4875269f44a701c5e58190a1d2f6fcb577ea79d842522dc
ab20ccb39b7ad2'
'DataScienceBowl\\stage1_train\\cb4df20a83b2f38b394c67f1d9d4aef29f9794d5345da35763
18374ec3a11490'
'DataScienceBowl\\stage1_train\\cbca32daaae36a872a11da4eaff65d1068ff3f154eedc9d3fc
0c214a4e5d32bd'
...
```

---

2. 移動させたいファイルが存在するか判定

Pythonでファイル、ディレクトリ（フォルダ）の存在確認 を参考に，以下のプログラムによって移動させたいファイルの存在確認を行う．

```
os.path.exists(file_path)

print(os.path.exists(file_path))
# True
print(not os.path.exists(file_path))
# False
print(os.path.exists(dir_path))
# True
```

3. ファイルを移動する

Pythonでファイル・ディレクトリを移動するshutil.move を参考に，以下のプログラムによってファイルを移動する．

```
src = os.path.join(img_dir_path, p)
dst = dst_img_path
shutil.move(src, dst)
```

ここで，第一引数に移動させたいファイルやディレクトリのパス、第二引数に移動先のディレクトリのパスを指定する．

---

masks ディレクトリ内の png 画像を別のディレクトリに移動する．

1. 移動したいファイルの拡張子を取得する

Pythonでパス文字列からファイル名・フォルダ名・拡張子を取得、結合 を参考に，以下のプログラムによってファイルの拡張子を取得する．

```
file_path = './dir/subdir/filename.ext'
_, ext = os.path.splitext(file_path) # ファイルの拡張子を取得
print(ext) # .ext
```

2. 移動したいファイルの名前を images のファイル名と同一にする

解凍した生データでは，`images` 内に存在した画像1枚に対して複数枚のマスク画像が存在するため，マスク画像のファイル名は `images` 内の画像名とは異なるラベルが与えられている．そのため，このままでは `images`内のファイル名をもとにして，その画像に対応するマスク画像を検索することができない．そこで，移動先では新しく `masks` 内に0から連続する整数で名付けられたディレクトリを作成し，その中に `images` 内のファイル名と同名に変換したマスク画像を配置していく．そこで，マスク画像のリネーム処理が必要となる．このリネーム処理は，os.renameでファイル名を変更する方法を解説！ を参考に以下のように作成した．

```
old_names = file_list # ディレクトリ内の現在のファイル名のリスト
new_name = file_name # 変換したい名前
if not (new_name in old_names): # 同名のファイルがない場合
  os.rename(old_name, new_name) # ファイルリネーム
else:  new_name = old_name
```

---

3. 作成するディレクトリ名に現在時刻を入れる

ディレクトリを作成する作業は，デバッグを続けていくと同名のディレクトリがすでに存在するために処理の途中で中断することがある．そのため，同名のディレクトリが作成されるのを防ぐために，名前に現在時刻を入れることにした．これは，Pythonにてファイル名に日付（今日の日付）や現在時刻を入れる方法 を参考に以下のように作成した．

```
import datetime

if os.path.exists(dst): # 存在する場合
  now = datetime.datetime.now()
  dst += now.strftime('%Y%m%d_%H%M%S') # 年月日_時分秒
os.mkdir(dst)
```

これで，『年月日_時分秒』が含まれた名前を持つディレクトリを自動的に作成することができるようになった．

```
class MoveFile:
    def __init__(self, parent_path):
        self.parent_path = parent_path
        self.org_path = os.path.join(self.parent_path, 'original')

    def unpack(self, file_name: str, create_dir: bool=True):
        zip_path = os.path.join(self.org_path, file_name + '.zip')
        unpack_path = self.parent_path
        if create_dir:
            try:
                unpack_path = os.path.join(self.parent_path, file_name)
                os.mkdir(unpack_path) # create directry
            except FileExistsError as e:
                print("ディレクトリ作成時にエラーが発生しました: ", e)
                return
        with zipfile.ZipFile(zip_path) as existing_zip:
            existing_zip.extractall(unpack_path)

    def _makedir(self, src: str, dir_name: str) -> str:
        dst = os.path.join(src, dir_name)
        if os.path.exists(dst): # 存在する場合
            now = datetime.datetime.now()
```

```
            dst += now.strftime('%Y%m%d_%H%M%S')
        os.mkdir(dst)
        return dst

    def MoveImage(self, src: str, dst: str):
        # 出力先のディレクトリが存在するか判定
        if not os.path.exists(dst): # 存在しない場合
            os.mkdir(dst)
        # 出力先の images ディレクトリを作成する
        dst_img_path = self._makedir(dst, 'images')
        # 出力先の masks ディレクトリを作成する
        dst_mask_path = self._makedir(dst, 'masks')

        # 全ての子ディレクトリの相対パスを取得 <Root> -> <0a7d330>
        list_dir_path = glob(os.path.join(src, '**'))
        for dir_path in list_dir_path:
            # 親ディレクトリ名を取得
            dir_name = os.path.basename(dir_path.rstrip(os.sep))
            # 子ディレクトリ内の images ディレクトリの相対パスを取得 <0a7d330> -> images
            list_img_dir_path = [f for f in glob(os.path.join(dir_path, '**')) if
'images' in f]
            for img_dir_path in list_img_dir_path:
                list_file = os.listdir(img_dir_path)
                if list_file: # ディレクトリ内にファイルが存在するか判定
                    for p in list_file:
                        # ディレクトリ内の全てのファイルを移動
                        shutil.move(os.path.join(img_dir_path, p), dst_img_path)
            # 子ディレクトリ内の masks ディレクトリの相対パスを取得 <0a7d330> -> masks
            list_masks_dir_path = [f for f in glob(os.path.join(dir_path, '**'))
if 'masks' in f]
            for masks_dir_path in list_masks_dir_path:
                list_file = os.listdir(masks_dir_path)
                if list_file: # ディレクトリ内にファイルが存在するか判定
                    for num, p in enumerate(list_file):
                        old_name = os.path.join(masks_dir_path, p)
                        # 拡張子のみ取得
                        _, ext = os.path.splitext(p)
                        new_name = os.path.join(masks_dir_path, dir_name+ext) #
C\\dataset\\masks\\label_name + \\label_name + .png

                        if not (p in new_name): # 同名のファイルがない場合
                            os.rename(old_name, new_name) # ファイルリネーム
                        else:  new_name = old_name

                        dst_path = os.path.join(dst_mask_path, str(num))
                        if not os.path.exists(dst_path):
                            os.mkdir(dst_path)
                        # ディレクトリ内のファイルを移動
                        shutil.move(new_name, dst_path)
```