from torchvision import transforms

class ImageTransform():

'''画像の前処理クラス。訓練時、検証時で異なる動作をする。

Attributes:

data\_transform(dic):

train: 訓練用のトランスフォーマーオブジェクト

val : 検証用のトランスフォーマーオブジェクト

'''

def \_\_init\_\_(self, resize, mean, std):

'''トランスフォーマーオブジェクトを生成する。

Parameters:

resize(int): リサイズ先の画像の大きさ

mean(tuple): (R, G, B)各色チャネルの平均値

std : (R, G, B)各色チャネルの標準偏差

'''

# dicに訓練用、検証用のトランスフォーマーを生成して格納

self.data\_transform = {

'train': transforms.Compose([

# ランダムにトリミングする

transforms.RandomResizedCrop(

resize, # トリミング後の出力サイズ

scale=(0.5, 1.0)), # スケールの変動幅

transforms.RandomHorizontalFlip(p = 0.5), # 0.5の確率で左右反転

transforms.RandomRotation(15), # 15度の範囲でランダムに回転

transforms.ToTensor(), # Tensorオブジェクトに変換

transforms.Normalize(mean, std) # 標準化

]),

'val': transforms.Compose([

transforms.Resize(resize), # リサイズ

transforms.CenterCrop(resize), # 画像中央をresize×resizeでトリミング

transforms.ToTensor(), # テンソルに変換

transforms.Normalize(mean, std) # 標準化

])

}

def \_\_call\_\_(self, img, phase='train'):

'''オブジェクト名でコールバックされる

Parameters:

img: 画像

phase(str): 'train'または'val' 前処理のモード

'''

return self.data\_transform[phase](img) # phaseはdictのキー