用語説明

マニュアルで使用されている用語の解説は次のとおりである。

用　　語	解説
オルソモザイク化	空中写真の像には位置ズレが生じる。正確な藻場面積の判定には写真上の像の位置ズレをなくし空中写真を地図と同じく、真上から見たような傾きのないものにする必要がある。標高データも用いて正しい大きさと位置に表示される画像に変換することオルソ化（補正）という。加えて、撮影した複数の画像を一括で処理したい場合、複数の写真画像を 1 枚の画像（モザイク処理）すること。両者を合わせてオルソモザイク化と呼ぶ。
幾何補正	オルソ化と同様の意味で、地図や画像などの空間位置の誤差を除き、基準点を用いて正確な地理座標を与えること。
教師	事前に与えられた現地データをいわば「例題（＝先生からの助言）」とみなすこと。
教師あり分類	事前に与えられたデータをいわば「例題（＝先生からの助言）」とみなして、それをガイドに藻場の分布などを推定する方法
検証	潜水観察や船上調査などの現地調査で得られる情報を用いて、他の調査（衛星画像、航空写真、ドローン空撮、音響測量）で得られた情報の正答率を算出すること。
深層学習	人間が自然に行うタスクをコンピュータに学習させる機械学習の手法の一つで、ニューラルネットワークを多層に結合して表現・学習能力を高めた手法である。近年、藻場の画像解析などにも使われ始めている。
振動子	音響測量に用いる魚群探知機などの超音波を発射して反射してきた音波を感知するセンサーのこと。
正答率（精度）	潜水観察や船上調査などの現地調査で得られる情報を用いて、他の調査（衛星画像、航空写真、ドローン空撮、音響測量）で得られた情報がどの程度あっているか算出した値。通常、70～80%程度の正答率があればよいとされる。
手引きの利用者	漁業者：漁業者、漁業協同組合、漁業者が組織する研究会等 行政機関：国、都道府県、市町村で試験研究機関を除いたもの 研究機関：大学、国・都道府県、独立行政法人等の試験研究機関 ＮＰＯ：藻場の保全回復活動を行う組織を言う。
バンド	衛星画像の光学センサーでは一つの機器で複数の波長帯を同時に観測でき、それぞれの波長帯毎に「バンド」という単位で分けられている。バンドとして可視光線（青バンド+緑バンド+赤バンドのBGRなど）と人間の視覚では捉えきれない近赤外線、赤外線の波長などがある。
藻　場	浅海域において海藻あるいは海草が繁茂している場所、あるいはそれらの群落群や群落内の動物を含めた群集のことをいう。

引用文献

1. 濵岡秀樹 (2020) 航空写真を用いた藻場判別における解析手法間での精度比較. 新水海研報 16:11-16
2. 国土地理院 オルソ画像について. https://www.gsi.go.jp/gazochosa/gazochosa40002.html
3. 国立研究開発法人水産研究・教育機構 水産工学研究所, 東北区水産研究所, 西海区水産研究所, 他(2020) 平成31年度水産基盤整備調査委託事業報告書 「藻場回復・保全技術の高度化検討調査」
4. Sonoki S, Shao H, Morita Y, et al (2016) Using acoustics to determine eelgrass bed distribution and to assess the seasonal variation of ecosystem service. PLoS One 11:1-15. https://doi.org/10.1371/journal.pone.0150890
5. 奥村宏征, 浅海茂, 森松秀治, 他 (2006) マルチビームソナーを用いたアマモ群落現存量推定法の開発. 海岸工学論文集 53:1436-1440
6. Shao H, Minami K, Shirakawa H, et al (2017) Verification of echosounder measurements of thickness and spatial distribution of kelp forests. J Mar Sci Technol 25:343-351. https://doi.org/10.6119/JMST-016-1229-1
7. 南憲吏, 安間洋樹, 濱野明, 他 (2015) 音響反射強度測定による来留見ノ瀬のホンダワラ科藻場 の現存量推定と季節変化. 海洋音響学会誌 42:1-10. https://doi.org/10.3135/jmasj.42.1
8. Abukawa K, Yamamuro M, Kikvidze Z, et al (2013) Assessing the biomass and distribution of submerged aquatic vegetation using multibeam echo sounding in Lake Towada, Japan. Limnology 14:39-42. https://doi.org/10.1007/s10201-012-0383-7
9. 梶原瑠美子, 桑原久実, 濱田保夫, 中嶋泰 (2015) 藻場や磯焼け域の把握に関わる新たな装置や技術～廉価版サイドスキャンソナー, ラジコンヘリ, 間欠撮影カメラの利用～. 水産工学 51:221-226
10. 水産工学研究所 (2011) 簡単に行える音響測器を用いた漁 場調査に関する手引き. http://nrife.fra.affrc.go.jp/topics/onnkyoukiki_tebiki/onnkyou_tebiki.pdf
11. Hamaoka H, Kamiyama T, Hori M (2020) Estimating the change in regional scale distribution of seagrass and macroalgal beds using discrete local distribution data analyzed from aerial images. Ecol Res 35:76-94. https://doi.org/10.1111/1440-1703.12087
12. Setyawidati NAR, Puspita M, Kaimuddin AH, et al (2018) Seasonal biomass and alginate stock assessment of three abundant genera of brown macroalgae using multispectral high resolution satellite remote sensing: A case study at Ekas Bay (Lombok, Indonesia). Mar Pollut Bull 131:40-48. https://doi.org/10.1016/j.marpolbul.2017.11.068
13. Vahtmae E, Kutser T (2007) Mapping Bottom Type and Water Depth in Shallow Coastal Waters with Satellite Remote Sensing. J Coast Res 185-189
14. Mumby PJ, Edwards AJ (2002) Mapping marine environments with IKONOS imagery: Enhanced spatial resolution can deliver greater thematic accuracy. Remote Sens Environ 82:248-257. https://doi.org/10.1016/S0034-4257(02)00041-X
15. 環境省 (2018) 瀬戸内海における藻場・干潟分布状況調査について. http://www.env.go.jp/water/heisa/survey/result_setonaikai.html

16. Cavanaugh KC, Siegel DA, Kinlan BP, Reed DC (2010) Scaling giant kelp field measurements to regional scales using satellite observations. Mar Ecol Prog Ser 403:13-27. https://doi.org/10.3354/meps08467
17. Wilson KL, Skinner MA, Lotze HK (2019) Eelgrass (Zostera marina) and benthic habitat mapping in Atlantic Canada using high-resolution SPOT 6/7 satellite imagery. Estuar Coast Shelf Sci 226:106292. https://doi.org/10.1016/j.ecss.2019.106292
18. Yamakita T, Watanabe K, Nakaoka M (2011) Asynchronous local dynamics contributes to stability of a seagrass bed in Tokyo Bay. Ecography (Cop) 34:519-528. https://doi.org/10.1111/j.1600-0587.2010.06490.x
19. Zavalas R, Ierodiaconou Daniel D, Ryan D, et al (2014) Habitat classification of temperate marine macroalgal communities using bathymetric LiDAR. Remote Sens 6:2154-2175. https://doi.org/10.3390/rs6032154
20. JAXA 衛星データを利用したカラー合成. https://www.eorc.jaxa.jp/hatoyama/experience/dataproc/
21. Lyzenga DR (1981) Remote sensing of bottom reflectance and water attenuation parameters in shallow water using aircraft and landsat data. Int J Remote Sens 2:71-82. https://doi.org/10.1080/01431168108948342
22. Sagawa T, Boisnier E, Komatsu T, et al (2010) Using bottom surface reflectance to map coastal marine areas: A new application method for Lyzenga's model. Int J Remote Sens 31:3051-3064. https://doi.org/10.1080/01431160903154341
23. Yamakita T, Sodeyama F, Whanpetch N, et al (2019) Application of deep learning techniques for determining the spatial extent and classification of seagrass beds, Trang, Thailand. Bot Mar 62:291-307. https://doi.org/10.1515/bot-2018-0017
24. Ishiguro S, Yamada K, Yamakita T, et al (2016) Classification of Seagrass Beds by Coupling Airborne LiDAR Bathymetry Data and Digital Aerial Photographs. 59-70. https://doi.org/10.1007/978-981-10-0780-4_5
25. 梶原瑠美子, 大橋正臣, 三上信雄, 他 (2016) 衛星画像による海底被覆物マップを用いた 漁場環境評価手法の検討. 土木学会論文集B3 (海洋開発) 72:
26. 京田潤一, 細川真也, 渡辺健太郎, 他 (2012) 現地観測データと衛星画像を用いた海草藻場の分布域と被度の推定. 土木学会論文集B2 (海岸工学) 68:I_1466-I_1470. https://doi.org/10.2208/kaigan.68.i_1466
27. 環境庁自然保護局 (1994) 第4回自然環境保全基礎調査海辺調査 総合報告書
28. 二宮順一, 森信人, 矢持進 (2006) 高解像度画像を用いた光学理論による藻場分布推定法の開発. 海岸工学論文集 53:1426-1430
29. Uhrin A V., Townsend PA (2016) Improved seagrass mapping using linear spectral unmixing of aerial photographs. Estuar Coast Shelf Sci 171:11-22. https://doi.org/10.1016/j.ecss.2016.01.021
30. Lathrop RG, Montesano P, Haag S (2013) A Multi-scale Segmentation Approach to Mapping Seagrass Habitats Using Airborne Digital Camera Imagery. Photogramm Eng Remote Sens 72:665-675. https://doi.org/10.14358/pers.72.6.665
31. 宇野女草太, 吉田夏樹, 高野正範, 他(2018) 航空機搭載型センサを用いた沿岸環

境調査事例の紹介. J Remote Sens Soc Japan 38:219-224
32. Zhao J, Zhao X, Zhang H, Zhou F (2017) Shallow water measurements using a single green laser corrected by building a near water surface penetration model. Remote Sens 9:1-18. https://doi.org/10.3390/rs9050426
33. Webster T, McGuigan K, Crowell N, et al (2016) Optimization of Data Collection and Refinement of Post-processing Techniques for Maritime Canada's First Shallow Water Topographic-bathymetric Lidar Survey. J Coast Res 76:31-43. https://doi.org/10.2112/SI76-004
34. Collin A, Long B, Archambault P (2012) Merging land-marine realms: Spatial patterns of seamless coastal habitats using a multispectral LiDAR. Remote Sens Environ 123:390-399. https://doi.org/10.1016/j.rse.2012.03.015
35. 国立研究開発法人水産研究・教育機構 水産工学研究所, 東北区水産研究所, 西海区水産研究所, 他(2020) 平成３１年度水産基盤整備調査委託事業報告書 「藻場回復・保全技術の高度化検討調査」
36. 中西達也, 棚田教生, 北野慎容, 他 (2019) 徳島県南部海部郡沿岸の2007年から2017年における藻場の変遷. 徳島県水産研究報告 34:1-34
37. 押野明夫, 斎藤憲次郎, 芳賀圭悟, 他 (2011) 宮城県北部岩礁域における藻場とキタムラサキウニの分布態様. 宮城県水産研究報告 11:43-64
38. Wada K (2016) labelme: Image Polygonal Annotation with Python
39. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp 234-241
40. Fujita D, Ishikawa T, Kodama S, et al (2006) Distribution and recent reduction of Gelidium beds in Toyama Bay, Japan. J Appl Phycol 18:591-598. https://doi.org/10.1007/s10811-006-9060-8

深層学習による水中画像（水中ドローン、潜水撮影画像）からの海藻判別の手順

・利用したシステム
Windows 10 pro 64 bit
プロセッサ：Intel® Core(TM) i7-8700 CPU@3.70GHz
実装メモリ (RAM) ：16.0 GB
GPU: NVIDIA GeForce GTX 1060 3GB
CUDA: Cuda version 10.1
conda version 4.9.2
python version 3.8.3
Spyder 4.1.4

手順
水中ドローンで撮影した動画から静止画を数秒おきに取り出す
（動画切り出し参考：https://note.nkmk.me/python-opencv-video-to-still-image/）

```
#python 動画から静止画切り出しプログラム
import cv2
import datetime
import os

os.chdir("動画のあるディレクトリへのパス")

# 時間（秒）で静止画切り出し範囲を指定する関数の作成--------------------------
--
def save_frame_range_sec(video_path, start_sec, stop_sec, step_sec,
                         dir_path, basename, ext='jpg'):
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        return

    os.makedirs(dir_path, exist_ok=True)
    base_path = os.path.join(dir_path, basename)

    digit = len(str(int(cap.get(cv2.CAP_PROP_FRAME_COUNT))))

    fps = cap.get(cv2.CAP_PROP_FPS)
    fps_inv = 1 / fps

    sec = start_sec
    while sec < stop_sec:
        n = round(fps * sec)
        cap.set(cv2.CAP_PROP_POS_FRAMES, n)
        ret, frame = cap.read()
        dt=datetime.datetime(2020, 10, 28, 10, 40, 22)#動画ファイルごとの開始時
刻を設定（年、月、日、時間、分、秒）
        td=datetime.timedelta(seconds=round(n * fps_inv))
        if ret:
            cv2.imwrite(
                '{}_{}_{}.{}'.format(
                    base_path, str(n).zfill(digit), (dt+td).strftime('%H%M%S'),
ext
```

```
                ),
                frame
            )
        else:
            return
        sec += step_sec
# 時間（秒）で範囲指定する関数の実行----
save_frame_range_sec(video_path, start_sec, stop_sec, step_sec,
                     dir_path, basename, ext='jpg')
# video_path は動画のあるディレクトリ、start_sec は切り抜きの開始時刻、stop_sec は切り抜きの終了時刻, step_sec は切り抜きの間隔（秒)、basename は切り抜き静止画につける共通名、ext='jpg'は静止画の拡張子の設定

save_frame_range_sec('C:/Python/video/xxx.mp4',
                     127, 1701, 1,
                     'C:/Python/yyyy','1028Iwafukaba')
#127 秒から 1701 秒後までで 1 秒ごとに静止画切り抜きする例
```

水中画像からのアノテーション（学習データ）の作成

水中ドローン静止画像および潜水撮影画像からアノテーションソフト labelme（https://github.com/hellock/icrawler）を用いて、画像上の海藻をアノテーション（画像上の分類したい物体を指定し、教師データを作る）し、このアノテーションデータを基に新規の画像に対して海藻（褐藻、紅藻、緑藻）の判別を行う。

anaconda prompt 上で lableme をインストールする。

(参考サイト：https://qiita.com/mucchyo/items/d21993abee5e6e44efad)

P45 のコラム 8 の図のように labelme を使うと、判別したい藻場部分を囲んでアノテーションデータ（教師データ）を作成できる。今回の深層学習の方法では作成したアノテーションデータセット（json ファイル）を PascalVOC 形式（アノテーションされた png ファイル）に変換しなくてはいけない。

labelme のソースから以下の PascalVOC 変換用のコードを手に入れて利用する。

labelme¥examples¥semantic_segmentation¥labelme2voc.py

以下を anaconda prompt 上で実行（labelme が最新バージョンであることを確認）。

```
python labelme2voc.py "dataset" "VOC2012" --labels labels.txt
```

labelme2voc.py [input_dir] [output_dir] --labels [labels]

input_dir: labelme でアノテーションした画像と json セットのディレクトリへのパス

output_dir: 出力ディレクトリへのパス

labels: ラベル情報テキストへのパス

labels.txt は labelme でアノテーションした分類名を同じように入れる（kasso を Kasso などにすると設定できない）

```
labels.txt の例
__ignore__
_background_
kasso
koso
ryokuso
```

Unet を使った深層学習による判別

VOC のフォルダの下に、深層学習用の該当フォルダを作ってその中に”train”フォルダと”test”フォルダを作る（図 S1）。”train”フォルダには images フォルダを作ってその中に train 用の jpg ファイルを入れ、もう一つ”seg_images”フォルダを作って、その中に VOC 変換で作成した SegmentationClassPNG フォルダ内の png ファイルを入れる。test フォルダ内に”images”フォルダを作ってその中に深層学習により判別する新規の jpeg 画像を入れる。

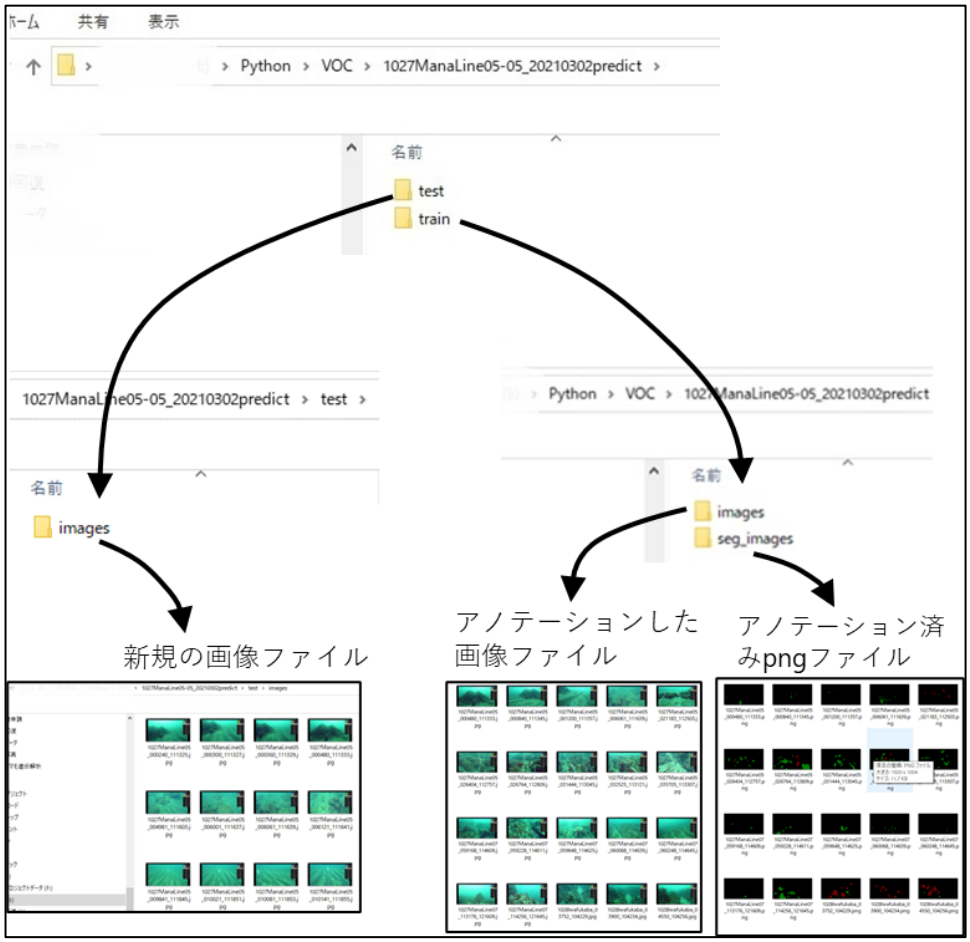

図 S1. Unet を使った深層学習の際の学習用の画像と判別する新規画像のフォルダ分け

spyder（python 分析向けの統合開発環境）を用いて、下記の python コードを実行し、深層学習による海藻判別を行う。

```
import os
import torch
import torch.nn.functional as F
import argparse
import cv2
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from copy import copy
from tqdm import tqdm

os.chdir("C:xxx")#パスの設定

# class config
```

```
class_label = {'koso': [0,0,128], 'kasso': [0,128,0], 'ryokuso':
[0,128,128]}
class_N = len(class_label) + 1 # class + background

# config
img_height, img_width = 256, 256 #572, 572
out_height, out_width = 256, 256 #388, 388
channel = 3

# GPU
GPU = "cuda" #or True CPU の場合は False
################################################
device = torch.device("cuda" if GPU and torch.cuda.is_available() else
"cpu")

# other
model_path = 'UNet1213.pt'

# random seed
torch.manual_seed(0)

class UNet(torch.nn.Module):
    def __init__(self):

        class UNet_block(torch.nn.Module):
            def __init__(self, dim1, dim2):
                super(UNet_block, self).__init__()

                _module = []

                for i in range(2):
                    f = dim1 if i == 0 else dim2
                    _module.append(torch.nn.Conv2d(f, dim2, kernel_size=3,
padding=1, stride=1))
                    _module.append(torch.nn.BatchNorm2d(dim2))
                    _module.append(torch.nn.ReLU())

                self.module = torch.nn.Sequential(*_module)

            def forward(self, x):
                x = self.module(x)
                return x

        class UNet_deconv_block(torch.nn.Module):
            def __init__(self, dim1, dim2):
                super(UNet_deconv_block, self).__init__()

                self.module = torch.nn.Sequential(
                    torch.nn.ConvTranspose2d(dim1, dim2, kernel_size=2,
stride=2),
```

```
                    torch.nn.BatchNorm2d(dim2)
                )

            def forward(self, x):
                x = self.module(x)
                return x

        super(UNet, self).__init__()

        base = 16

        self.enc1 = UNet_block(3, base)
        self.enc2 = UNet_block(base, base * 2)
        self.enc3 = UNet_block(base * 2, base * 4)
        self.enc4 = UNet_block(base * 4, base * 8)
        self.enc5 = UNet_block(base * 8, base * 16)

        self.tconv4 = UNet_deconv_block(base * 16, base * 8)
        self.tconv3 = UNet_deconv_block(base * 8, base * 4)
        self.tconv2 = UNet_deconv_block(base * 4, base * 2)
        self.tconv1 = UNet_deconv_block(base * 2, base)

        self.dec4 = UNet_block(base * 16, base * 8)
        self.dec3 = UNet_block(base * 8, base * 4)
        self.dec2 = UNet_block(base * 4, base * 2)
        self.dec1 = UNet_block(base * 2, base)

        self.out = torch.nn.Conv2d(base, class_N, kernel_size=1, padding=0,
stride=1)


    def forward(self, x):
        # block conv1
        x_enc1 = self.enc1(x)
        x = F.max_pool2d(x_enc1, 2, stride=2, padding=0)

        # block conv2
        x_enc2 = self.enc2(x)
        x = F.max_pool2d(x_enc2, 2, stride=2, padding=0)

        # block conv31
        x_enc3 = self.enc3(x)
        x = F.max_pool2d(x_enc3, 2, stride=2, padding=0)

        # block conv4
        x_enc4 = self.enc4(x)
        x = F.max_pool2d(x_enc4, 2, stride=2, padding=0)

        # block conv5
        x = self.enc5(x)
```

```
        x = self.tconv4(x)

        x = torch.cat((x, x_enc4), dim=1)
        x = self.dec4(x)

        x = self.tconv3(x)

        x = torch.cat((x, x_enc3), dim=1)
        x = self.dec3(x)

        x = self.tconv2(x)
        x = torch.cat((x, x_enc2), dim=1)
        x = self.dec2(x)

        x = self.tconv1(x)
        x = torch.cat((x, x_enc1), dim=1)
        x = self.dec1(x)

        x = self.out(x)
        x = F.softmax(x, dim=1)

        return x

# get train data
def data_load(path, hf=False, vf=False, rot=False):
    if (rot == 0) and (rot != False):
        raise Exception('invalid rot >> ', rot, 'should be [1, 359] or
False')

    paths = []
    paths_gt = []

    data_num = 0
    for dir_path in glob(path + '*'):
        data_num += len(glob(dir_path + "*"))

    pbar = tqdm(total = data_num)

    for dir_path in glob(path + '*'):
        for path in glob(dir_path + '*'):
            for i, cls in enumerate(class_label):
                if cls in path:
                    t = i

            paths.append({'path': path, 'hf': False, 'vf': False, 'rot':
0})

            gt_path = path.replace("images", "seg_images").replace(".jpg",
".png")
```

```
            paths_gt.append({'path': gt_path, 'hf': False, 'vf': False,
'rot': 0})

            # horizontal flip
            if hf:
                paths.append({'path': path, 'hf': True, 'vf': False, 'rot':
0})
                paths_gt.append({'path': gt_path, 'hf': True, 'vf': False,
'rot': 0})
            # vertical flip
            if vf:
                paths.append({'path': path, 'hf': False, 'vf': True, 'rot':
0})
                paths_gt.append({'path': gt_path, 'hf': False, 'vf': True,
'rot': 0})
            # horizontal and vertical flip
            if hf and vf:
                paths.append({'path': path, 'hf': True, 'vf': True, 'rot':
0})
                paths_gt.append({'path': gt_path, 'hf': True, 'vf': True,
'rot': 0})
            # rotation
            if rot is not False:
                angle = rot
                while angle < 360:
                    paths.append({'path': path, 'hf': False, 'vf': False,
'rot': rot})
                    paths_gt.append({'path': gt_path, 'hf': False, 'vf':
False, 'rot': rot})
                    angle += rot

            pbar.update(1)

    pbar.close()

    return np.array(paths), np.array(paths_gt)

def get_image(infos, gt=False):
    xs = []

    for info in infos:
        path = info['path']
        hf = info['hf']
        vf = info['vf']
        rot = info['rot']
        x = cv2.imread(path)

        # resize
        if gt:
            x = cv2.resize(x, (img_width, img_height)).astype(np.float32)
```

```
        else:
            x = cv2.resize(x, (out_width, out_height)).astype(np.float32)

        # channel BGR -> Gray
        if channel == 1:
            x = cv2.cvtColor(x, cv2.COLOR_BGR2GRAY)
            x = np.expand_dims(x, axis=-1)

        # horizontal flip
        if hf:
            x = x[:, ::-1]

        # vertical flip
        if vf:
            x = x[::-1]

        # rotation
        scale = 1
        _h, _w, _c = x.shape
        max_side = max(_h, _w)
        tmp = np.zeros((max_side, max_side, _c))
        tx = int((max_side - _w) / 2)
        ty = int((max_side - _h) / 2)
        tmp[ty: ty+_h, tx: tx+_w] = x.copy()
        M = cv2.getRotationMatrix2D((max_side / 2, max_side / 2), rot,
scale)
        _x = cv2.warpAffine(tmp, M, (max_side, max_side))
        x = _x[tx:tx+_w, ty:ty+_h]

        if gt:
            _x = x
            x = np.zeros((out_height, out_width), dtype=np.int)

            for i, (_, vs) in enumerate(class_label.items()):
                ind = (_x[..., 0] == vs[0]) * (_x[..., 1] == vs[1]) *
(_x[..., 2] == vs[2])
                x[ind] = i + 1
        else:
            # normalization [0, 255] -> [-1, 1]
            x = x / 127.5 - 1

            # channel BGR -> RGB
            if channel == 3:
                x = x[..., ::-1]

        xs.append(x)

    xs = np.array(xs, dtype=np.float32)

    if not gt:
```

```
        xs = np.transpose(xs, (0,3,1,2))

    return xs


# train
def train():
    # model
    model = UNet().to(device)
    model.train()

    opt = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

    paths, paths_gt = data_load('./XXX/xxx/train/images/', hf=True,
vf=True, rot=False)    ####### train データフォルダ

    # training
    mb = 8 # batch size
    mbi = 0
    train_N = len(paths)
    train_ind = np.arange(train_N)
    np.random.seed(0)
    np.random.shuffle(train_ind)

    loss_fn = torch.nn.NLLLoss()

    for i in range(1000):
        if mbi + mb > train_N:
            mb_ind = copy(train_ind[mbi:])
            np.random.shuffle(train_ind)
            mb_ind = np.hstack((mb_ind, train_ind[ : (mb - (train_N -
mbi))]))
            mbi = mb - (train_N - mbi)
        else:
            mb_ind = train_ind[mbi : mbi + mb]
            mbi += mb

        # data load
        x = torch.tensor(get_image(paths[mb_ind]),
dtype=torch.float).to(device)
        t = torch.tensor(get_image(paths_gt[mb_ind], gt=True),
dtype=torch.long).to(device)

        opt.zero_grad()
        y = model(x)

        # reshape gt
        y = y.permute(0, 2, 3, 1).contiguous()
        y = y.view(-1, class_N)
        t = t.view(-1)
```

```
        loss = loss_fn(torch.log(y), t)
        loss.backward()
        opt.step()

        pred = y.argmax(dim=1, keepdim=True)
        acc = pred.eq(t.view_as(pred)).sum().item() / mb / out_height /
out_width

        if (i + 1) % 10 == 0:
            print('Iter : {} , Loss : {} , Accuracy : {}'.format(i + 1,
loss.item(), acc))

    torch.save(model.state_dict(), model_path)

# test
def test():
    model = UNet().to(device)
    model.load_state_dict(torch.load(model_path,
map_location=torch.device(device)))
    model.eval()

    paths, path_gt = data_load('./XXX/xxx /test/images/')   ####### testデ
ータフォルダ

    with torch.no_grad():
        for i in range(len(paths)):
            path = paths[[i]]
            x = get_image(path)

            x = torch.tensor(get_image(paths[[i]]),
dtype=torch.float).to(device)

            pred = model(x)

            #pred = pred.permute(0,2,3,1).reshape(-1, class_num+1)
            pred = pred.detach().cpu().numpy()[0]
            pred = pred.argmax(axis=0)

            # prediction -> RGB
            out = np.zeros((out_height, out_width, 3), dtype=np.uint8)
            for i, (_, vs) in enumerate(class_label.items()):
                out[pred == (i + 1)] = vs

            print(">> {}".format(path[0]['path']))

            plt.subplot(1, 2, 1)
            plt.imshow((x.detach().cpu().numpy()[0].transpose(1,2,0) *
127.5 + 127.5).astype(np.uint8))
            plt.subplot(1, 2, 2)
```

```
            plt.imshow(out[..., ::-1])
            plt.show()


def arg_parse():
    parser = argparse.ArgumentParser(description='')
    parser.add_argument('--train', default='--train', dest='train',
action='store_true')
    parser.add_argument('--test',  default='--test',  dest='test',
action='store_true')
    args = parser.parse_args()
    return args

# main
if __name__ == '__main__':
    args = arg_parse()

    if args.train:
        train()
    if args.test:
        test()

    if not (args.train or args.test):
        print("please select train or test flag")
        print("train: python main.py --train")
        print("test:  python main.py --test")
        print("both:  python main.py --train --test")
```

このコードを spyder で実行すると、spyder のプロットペイン（右上）で元画像と判別結果の画像が出てくるので保存する。プロットペイン内ですべての結果を画像として一括保存できる（図 S2）。ただし、ファイル名が保存した時間になってしまい（例. Figure 2021-03-02 151522 (1).png）、元ファイルとの紐づけがなくなってしまう。元ファイル名への修正方法については後述する。

・エラーが出た時の対応
RuntimeError: CUDA out of memory. Tried to allocate 16.00 MiB (GPU 0; 3.00 GiB total capacity; 1.94 GiB already allocated; 11.86 MiB free; 1.97 GiB reserved in total by PyTorch)
→上記のコード上でバッチサイズを設定できるので、サイズを小さくして対応する。
mb=16 to mb=8

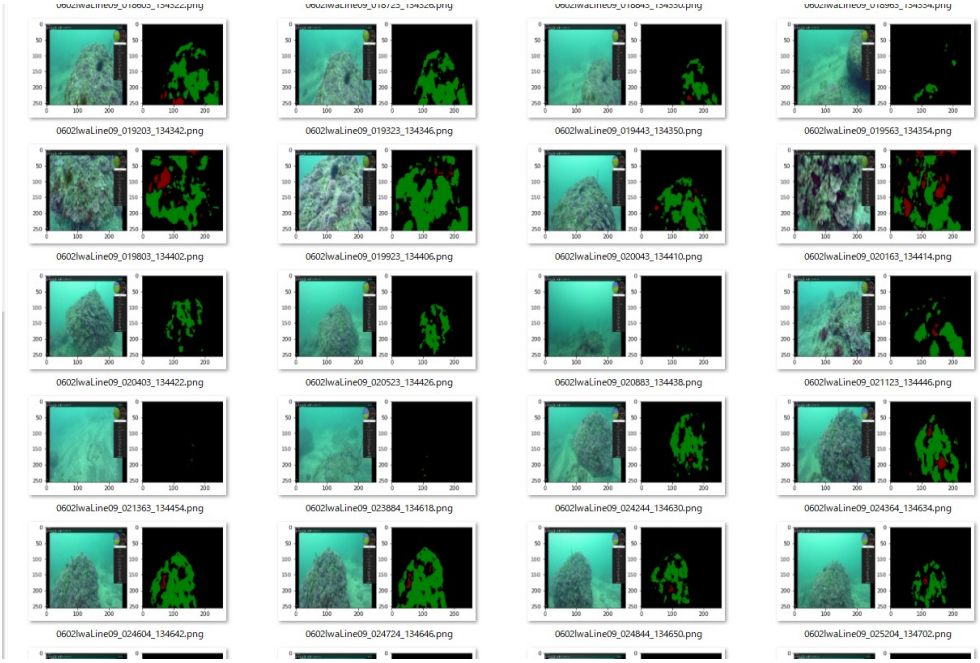

図 S2. Unet を使った深層学習の海藻判別結果．元画像と判別画像が結合されている

判別結果からの被度の算出

・ファイル名の一括修正

深層学習の segmentation 結果の png ファイルを一括保存すると、元ファイルの名前から変わり、上記のようにファイル名が保存した時間になる。

うまく segmentation 結果に元ファイル名が付くようにするには以下の手順を踏む。

ファイル名をエクスプローラーの shift+右クリック→パスのコピーで一括コピーする。同様に変更先のファイル名（元のファイル名）もこの方法で一括コピーする。この時両者の順番が合うように画像を確認する。エクセルなどにコマンドプロンプトの名前変更コマンドの ren　*元ファイル名*　変更ファイル名を列に並べる。*で元ファイル名を囲んだのは元ファイル名の中に半角スペースがあって、そのままだとコマンドプロンプトでコマンドを読めないためである。メモ帳にコピーペーストし、その後″（ダブルクオーテーション）に変更し、列間のタブを半角スペースに置換する。（エクセルでは″（ダブルクオーテーション）は使えないので、代わりに*を使っている）

例. txt ファイルにコピペしたファイル名変更のコード

```
ren *Figure 2021-02-26 191002 (0).png* 1027ManaLine05_000240_11132.png
ren *Figure 2021-02-26 191002 (1).png* 1027ManaLine05_000300_11132.png
ren *Figure 2021-02-26 191002 (2).png* 1027ManaLine05_000360_11132.png
```

これをコマンドプロンプトにコピーしてすべてのファイル名を元ファイル名にする。

・判別画像のクリップ（抜き出し）

判別結果は元画像と海藻判別画像が結合されている（図 S2）ため、そこから判別（segmentation）部分だけを下記の python コードを用いて抜き出す。

```
import os
import glob
from PIL import Image

os.chdir("C:/Python/xxxx/xxxx ")#ディレクトリのパスを設定
```

```
def imgcrop(img):
    yield img.crop((216, 41, 345, 163)) # yield 関数の処理を一旦停止して、
戻り値を返す画像中の 切り出す領域を引数 box=(left, upper, right, lower)で設
定。

files = glob.glob('*.png')#ディレクトリ中のファイルを取得
for x in files:
    img = Image.open(x)
    for ig in imgcrop(img) :
        # 保存先フォルダの指定
        ig.save("C:/Python/xxx/xxxx" +x+"")# yield +x+はファイル名
```

・被度の計算

クリップした判別領域から BGR で褐藻（0,0,128）、紅藻(0,128,0)、緑藻(0.128,128)の色ごとにセグメンテーションされているので（図 S3）、下記の python コードでその色ごとに被度を計算すし、txt ファイルに保存する。

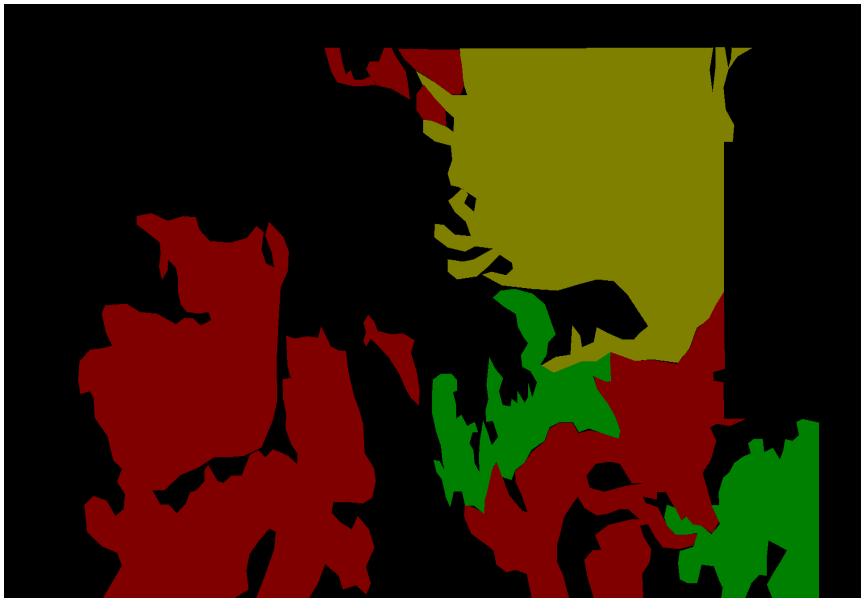

図 S3.海藻判別のセグメンテーション結果（褐藻が赤色、紅藻が緑色、緑藻が黄土色、裸地が黒色）

```
#海藻の被度計算のための 3 値画像（褐藻、紅藻、緑藻、）からの被度算出
import os
import cv2
import glob

os.chdir("C:/Python/xxxx/xxxx")
#切り抜き画像のディレクトリのパスに設定へ

#被度計算プログラム
def coverage_cal(img):
    height=122#画像の縦の画素数を任意に設定
    width=129 #画像の横の画素数を任意に設定
    whole_area=15738#全体の画素数（縦の画素数×横の画素数）
    rachi_area=0 #裸地の面積(初期=0)
```

```
    koso_area=0#紅藻(緑色) の面積(初期=0)
    kasso_area=0 #褐藻(赤色) の面積(初期=0)
    ryokuso_area=0#緑藻 (黄緑色) の面積(初期=0)
    other_area=0
    #各色の面積比を求める
    for i in range(height):
        for j in range(width):
            if all(img[i,j]<[5,5,5]):
                rachi_area+=1 #裸地
            elif all([0,5,0]<= img[i,j]) and all(img[i,j]<=[0,128,0]):
                koso_area+=1 # (BGR)の緑は紅藻
            elif all([0,0,5]<= img[i,j]) and all(img[i,j]<=[0,0,128]):
                kasso_area+=1 #(BGR)の赤は褐藻
            elif all([0,5,5]<= img[i,j]) and all(img[i,j]<=[0,128,128]):
                ryokuso_area+=1 #(BGR)の黄緑は緑藻
            else :
                other_area+=1 # その他
    kasso=str(kasso_area/whole_area*100)#褐藻
    koso=str(koso_area/whole_area*100)#紅藻
    ryokuso=str(ryokuso_area/whole_area*100)#緑藻
    other=str(other_area/whole_area*100)#その他
    rachi=str(rachi_area/whole_area*100)#裸地ド
    with open('results00.txt', 'a') as f:#"a"は追記モード、"w"は書き込みモ
ード
        print(x,kasso,koso,ryokuso,other,rachi, file=f)
    with open('results00.txt') as f:
        print(f.readlines())


files = glob.glob('*.png')#ディレクトリ中のファイルを取得
for x in files:
    img=cv2.imread(x)#画像ファイル読み込み
    coverage_cal(img)
```