高齢女性の下腿のデジタル画像による低骨格筋肉量判定の可能性

中口拓真（PT）[1)]，石本泰星（PT）[2)]，桑田一記（PT）[1)]，福本祐真（PT）[1)]，田津原佑介（PT）[1)]，近藤義剛（PT）[1)]

[1)] 貴志川リハビリテーション病院
[2)] 赤ひげクリニック

キーワード：高齢者，骨格筋量，機械学習

はじめに

高齢期にみられる骨格筋量，筋力，身体機能の低下はサルコペニアと呼ばれ[1)]，骨折リスクの増加[2)3)]，心疾患[4)]，呼吸器疾患[5)]，日常生活能力の低下[6)]，生活の質の低下[7)]および生命予後[8)]との関連が報告されている。また，高齢者においては，外科的手術の前にサルコペニアの評価をすることが重要であり[9)]，サルコペニアは術後の合併症を予測する[10)]。

サルコペニアの診断においてアジア作業グループ（Asian working group for sarcopenia；以下，AWGS）[1)]は地域在住高齢者のためのサルコペニア診断基準を発表しており，握力や歩行速度と併せて骨格筋指数（Skeletal Muscle Index；以下，SMI）を計測することでサルコペニア診断することを推奨している。SMI の計測にはコンピュータ断層撮影[11)]，生体電気インピーダンス法[12)]，などが存在する。これらは，特殊な機器を必要とし，運搬においても労力を有することから，実際の臨床現場において広く用いられているとは言い難い。下腿周径によるスクリーニング検査も報告されているが，人が接触する検査法であるため，新型コロナウイルスなどの感染リスクが存在するため，非接触型の検査法が望ましい。そのため，本研究では，下腿の画像を使用して，低 SMI を予測できるか予備的に検証した。

対象および方法

1．対象

対象は，2020 年 1 月 1 日～ 6 月 30 日まで貴志川リハビリテーション病院の回復期リハビリテーション病棟に入院していた 65 歳以上の女性とした。除外基準は，神経学的疾患を有する者，下腿に圧痕を残す浮腫のある者，BMI 25 kg/m^2 以上（肥満）である者，立位保持が不可能な者とした。本研究は，貴志川リハビリテーション病院の倫理委員会の承認（承認番号：4）を得ている。

2．調査・測定項目

1）対象者属性

対象者の属性として，年齢，身長，体重，BMI，疾患名を診療録から調査した。歩行能力の指標として歩行速度，下肢筋力の指標として膝伸展筋力，上肢筋力の指標として握力を調査した。

2）SMI

SMI の計測は，BIA 法（Bioelectrical Impedance Analysis：以下，BIA 法）で計測した。計測は排尿後，運動前に統一し，計測時間は午前 9 ～ 11 時の間に実施した。SMI の算出法は，四肢骨格筋量を身長の二乗で除した値を SMI と定義した。AWGS における女性のサルコペニアの基準である 5.7 kg/m^2 未満を低 SMI 群，それ以外を正常群とした。

3）下腿のデジタル画像

下肢に整形外科的疾患を有する場合は健側下腿外側の画像とし，下肢に整形外科的疾患を有さない場合や両側下肢に整形外科的疾患を有する場合は，膝伸展筋力の強い側を撮影した。対象者は裸足となり，膝関節から遠位の皮膚が観察できるように衣服を調整した。次に，対象者は昇降台上で端座位をとり，足底が床に接触した状態で下腿が床面に対し垂直となるように座面を調整した。撮影者は対象者の非撮影側下肢ができる限りカメラに映らないように，膝屈曲位で保持させた。撮影時の注意点として，カメラレンズの中央が撮影側下腿の腓骨頭と外果の中央で撮影側の腓骨と垂直になるように目視で調整し，下腿とカメラのレンズとの距離は 50 cm として非伸縮性のメジャーで距離を計測した。デジタルカメラは三脚に固定し三脚に取り付けられた水準器で水平を確認した後に撮影をした。また，デジタル画像を撮影する際は，窓のブラインドを下ろし，外の光が入らないようにした。撮影場所については，リハビリテーション室内で実施し，下腿以外の周辺環境が撮影者や対象者によって異ならないようにすべて同じ環境で撮影した。撮影にはデジタルカメラを使用した。

3．データ解析

すべての対象者属性データと SMI について平均値と標準偏差を調査した。低 SMI 判定についてのディープラーニングフレームワークは，先行研究でも使用していた Residual Network（以下，ResNet）とした。ResNet の出力層は正常群，低 SMI 群と 2 値化した。最適化手法は確率的勾配降下法，バッチサイズは 32 とした。活性化関数には Rectified Linear 関数，損失関数はクロスエントロピーとした。

すべてのデータを教師データと検証用データ，独立したテストデータにランダムに分け，テストデータを 30% とした。教師データの学習回数は 10 回として予測モデルを作成し，テストデータの予測モデルにおける C 統計量，感度，特異度を求めた。解析にはプログラミング言語 Python の PyTorch を使用し，C 統計量，感度，特異度は統計ソフト R 4.0 を使用した。本研究で用いた解析コードを Appendix にて公開する。

表1 対象者属性

項目	正常群（n=26）	低SMI群（n=26）
年齢（year）	75（10.6）	80（6.8）
身長（cm）	153.3（0.6）	146.3（0.6）
体重（kg）	53.5（4.1）	42.4（5.27）
BMI（kg/m^2）	22.8（1.4）	19.6（2.1）
SMI（kg/m^2）	5.78（0.3）	4.76（0.4）
歩行速度（m/sec）	1.07（0.39）	0.84（0.34）
膝伸展筋力（kgf/weight）	0.33（0.09）	0.31（0.07）
握力（kg）	18.3（5.97）	14.7（7.1）

平均値（標準偏差）
BMI：Body Mass Index
SMI：Skeletal Muscle Index

結　　果

研究期間内に52名（正常群26名，低SMI群26名）の同意が得られた。対象者の基本属性を表1に示す。デジタル画像による低SMIを判定するテストデータの低SMIの判定精度［推定値（95%信頼区間）］は，C統計量0.91（0.83–1.00），感度0.93（0.70–0.97），特異度0.90（0.70–0.97）であった。

考　　察

入院中の運動器疾患を有する高齢女性を対象に，AWGSの低SMI基準である5.7 kg/m^{2}未満を低SMI群として，機械学習のアルゴリズムであるResNetにより下腿の画像を用いた予測精度の判定を行った。その結果，C統計量，感度，特異度は高くスクリーニング検査として使用できる可能性を示した。

機械学習の予測モデルに関して，教師データでは精度が高く検証用データの精度を低下させる過学習が懸念されている。本研究ではサンプルサイズが小さいため過学習が発生しやすく，結果の解釈は慎重に行う必要がある。

また，先行研究[13)14)]では，SMIではなくサルコペニアを予測している。そのため，サルコペニア診断基準である歩行速度と握力を調査した。対象者属性において，AWGSでは，0.8 m/sec未満がサルコペニアを診断する基準のひとつであり，本研究では正常群と低SMI群の両方でAWGSの基準を上回っていた。また，握力に関してはAWGSの診断基準は，18 kg未満である。本研究では，正常群が18.3 kgとAWGSの基準をわずかに上回っているが，両群とも握力が低下した対象者であった。膝関節伸展筋力に関しては，両群ともに低値であった。本結果は上記のような身体機能を有する入院中の女性運動器疾患患者に適応できるものであるため，その他への一般化は現状困難である。よって今後は様々な疾患や地域在住高齢者を対象とした調査が望まれる。

本研究では，いくつかの限界がある。機械学習のアルゴリズムを使用した結果であるため，回帰式のように解析した時点ですぐに予測として使用できない点である。この点については，ウェブアプリケーションなどを開発することで容易に使用可能になると考える。2つめとして，本研究における対象者は入院中の運動器疾患患者であり，今後は入院歴がない地域在住高齢者を対象とした研究の実施が必要であると考える。最後に，本研究では下腿のデジタル画像と下腿周径における予測精度の比較は行えていないため，検証する必要がある。

結　　論

本研究では，入院中の運動器疾患を有する高齢女性に対し低SMIの判定を下腿のデジタル画像を用いて行った。その結果，低SMIを高い精度で判定できる可能性を示した。しかし，サンプルサイズが小さいことや理想条件のみでの検証であり，解釈には注意が必要である。

文　　献

1) Chen LK, Woo J, *et al.*: Asian Working Group for Sarcopenia: 2019 Consensus Update on Sarcopenia Diagnosis and Treatment. J Am Med Dir Assoc. 2020; 21: 300–307.
2) Bischoff-Ferrari HA, Orav JE, *et al.*: Comparative performance of current definitions of sarcopenia against the prospective incidence of falls among community-dwelling seniors age 65 and older. Osteoporos Int. 2015; 26: 2793–2802.
3) Schaap LA, Schoor MN, *et al.*: Associations of Sarcopenia Definitions, and Their Components, With the Incidence of Recurrent Falling and Fractures: The Longitudinal Aging Study Amsterdam. J Gerontol A Biol Sci Med Sci. 2018; 10: 1199–1204.
4) Bahat G, IIhan B, *et al.*: Sarcopenia and the cardiometabolic syndrome: A narrative review. Eur Geriatr Med. 2016; 7: 220–223.
5) Bone AE, Hepgul N, *et al.*: Sarcopenia and frailty in chronic respiratory disease. Chron Respir Dis. 2017; 14: 85–99.
6) Malmstrom TK, Miller DK, *et al.*: SARC-F: a symptom score to predict persons with sarcopenia at risk for poor functional outcomes. J Cachexia Sarcopenia Muscle. 2016; 7: 28–36.

7) Beaudart C, Emmanuel B, *et al*.: Validation of the SarQoL, a specific health-related quality of life questionnaire for Sarcopenia. J Cachexia Sarcopenia Muscle. 2017; 8: 238–244.
8) Stefanie L, Mirko P, *et al*.: Validation of the FNIH sarcopenia criteria and SOF frailty index as predictors of long-term mortality in ambulatory older men. Age Ageing. 2016; 45: 602–608.
9) Riccardo A, Barbara Le: When Reporting on Older Patients With Cancer, Frailty Information Is Needed. Ann Surg Oncol. 2011; 18: 4–5.
10) Vicente V, Neda A, *et al*.: Sarcopenia Adversely Impacts Postoperative Complications Following Resection or Transplantation in Patients With Primary Liver Tumors. J Gastrointest Surg. 2015; 19: 272–281.
11) Goodpaster BH, Kelley DE, *et al*.: Skeletal muscle attenuation determined by computed tomography is associated with skeletal muscle lipid content. J Appl Physiol. 2000; 1: 104–110.
12) Giuseppe S, Marina D, *et al*.: Measurement of lean body mass using bioelectrical impedance analysis: a consideration of the pros and cons. Aging Clin Exp Res. 2017; 29: 591–597.
13) Yang M, Xiaoyi H, *et al*.: SARC-F for Sarcopenia Screening in Community-Dwelling Older Adults: Are 3 Items Enough? Medicine. 2018; 9730: 11726.
14) Yang M, Xiaoyi H, *et al*.: Screening Sarcopenia in Community-Dwelling Older Adults: SARC-F vs SARC-F Combined With Calf Circumference (SARC-CalF). J Am Med Dir Assoc. 2018; 19: 277.e1–277.e8.

発表実績

【原著論文】

1) 中口拓真，石本泰星，他：高齢女性の下腿のデジタル画像による低骨格筋肉量判定の可能性― Convolutional Neural Network とエッジ検出を用いた分類による予備的研究―. 理学療法学. 2021; 48(3): 279–286.

Appendix

```
import os
import time
import copy

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, random_split
from torchvision import datasets, models, transforms
import optuna

import matplotlib.pyplot as plt
from cnn_finetune import make_model
import csv
import calc_img

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

def plot_result(phase,x,y_tloss,y_tacc,y_eloss,y_eacc):
    plt.clf()
    plt.subplot(211)
    plt.plot(x,y_tloss,"-",color="b",label='Train')
    plt.plot(x,y_eloss,"-",color="r",label='val')
    plt.title("Loss")
    plt.legend()

    plt.subplot(212)
    plt.plot(x,y_tacc,"-",color="b",label='Train')
    plt.plot(x,y_eacc,"-",color="r",label='val')
    plt.title("Accuracy")
    plt.legend()

    plt.tight_layout()
    plt.draw()
    return

def prepare_dataset(dataset, val_split=0.3):
    print(dataset.class_to_idx)
    dataset_size = len(dataset)
    train_size = int(dataset_size * 0.6)
    val_size = int(dataset_size * 0.2)
    test_size = dataset_size - train_size - val_size
    print('train size: ' + str(train_size))
    print('val size: ' + str(val_size))
```

```
    print('test size: ' + str(test_size))
    train, val, test = random_split(dataset, [train_size, val_size, test_size])

    change_list = [transforms.RandomHorizontalFlip(p=0.5),
                   transforms.RandomVerticalFlip(p=0.5)]

    calc_mean_std_class = calc_img.calc_mean_std()
    mean, std = calc_mean_std_class.calc(train, BATCH_SIZE, NUM_WORKER)
    if not GRAYSCALE:
        print('input images are RGB !')
        data_transforms = {
            'train': transforms.Compose([
                transforms.RandomChoice(change_list),
                transforms.ToTensor(),
                transforms.RandomErasing(p=0.3,scale=(0.02,0.33),ratio=(0.3,3.3)),
                transforms.Normalize(mean,std)
            ]),
            'val': transforms.Compose([
                transforms.ToTensor(),
                transforms.Normalize(mean,std)
            ]),
            'test':transforms.Compose([
                transforms.ToTensor(),
                transforms.Normalize(mean, std)
            ])
        }
    else:
        print('input images are GrayScale !')
        data_transforms = {
            'train': transforms.Compose([
                transforms.Grayscale(num_output_channels=3),
                transforms.RandomChoice(change_list),
                transforms.ToTensor(),
                transforms.RandomErasing(p=0.3,scale=(0.02,0.33),ratio=(0.3,3.3)),
                transforms.Normalize(mean,std)
            ]),
            'val': transforms.Compose([
                transforms.Grayscale(num_output_channels=3),
                transforms.ToTensor(),
                transforms.Normalize(mean,std)
            ]),
            'test':transforms.Compose([
                transforms.Grayscale(num_output_channels=3),
                transforms.ToTensor(),
                transforms.Normalize(mean, std)
            ])
        }

    train.dataset.transform = data_transforms['train']
    val.dataset.transform = data_transforms['val']
    test.dataset.transform = data_transforms['test']

    return {'train':train, 'val':val, 'test':test}

def create_dataloader(image_datasets):
    train_dataset_loader = DataLoader(
        image_datasets['train'], batch_size=BATCH_SIZE,
        shuffle=True, num_workers=NUM_WORKER
    )
    valid_dataset_loader = DataLoader(
        image_datasets['val'], batch_size=BATCH_SIZE,
        shuffle=True, num_workers=NUM_WORKER
    )
    test_dataset_loader = DataLoader(
        image_datasets['test'], batch_size=BATCH_SIZE,
        shuffle=False, num_workers=NUM_WORKER
    )
    dataloaders_comb = {
        "train": train_dataset_loader,
        "val": valid_dataset_loader,
```

```
        "test": test_dataset_loader,
    }
    return dataloaders_comb

def train_model(model, criterion, optimizer, num_trials, num_epochs=25):
    since = time.time()

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    x=list(range(num_epochs))
    y_tloss=[]; y_tacc=[]
    y_eloss=[]; y_eacc=[]

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch + 1, num_epochs))
        print('-' * 10)

        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()
            else:
                model.eval()

            running_loss = 0.0
            running_corrects = 0

            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                optimizer.zero_grad()

                # forward
                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)
                    # backward + optimize only if in training phase
                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]

            print('{} Loss: {:.4f} Acc: {:.4f}'.format(
                phase, epoch_loss, epoch_acc))

            if phase == 'train':
                y_tloss.append(epoch_loss)
                y_tacc.append(epoch_acc)
            else:
                y_eloss.append(epoch_loss)
                y_eacc.append(epoch_acc)
                if epoch == num_epochs-1:
                    fig, axes = plt.subplots(2,1)
                    plot_result(phase,x[:epoch+1],y_tloss,y_tacc,y_eloss,y_eacc)
                    fig.savefig("res"+str(num_trials)+".png")

                if epoch_acc > best_acc:
                    best_acc = epoch_acc
                    best_model_wts = copy.deepcopy(model.state_dict())

    time_elapsed = time.time() - since
    print('Training completed in {:.0f}m {:.0f}s'.format(time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))
```

```
    model.load_state_dict(best_model_wts)
    return model, best_acc, fig

def test_model(model):
    print("=" * 10)
    model.eval()   # Set model to evaluate mode

    running_corrects = 0
    for i, (inputs, labels) in enumerate(dataloaders['test'],0):
        inputs = inputs.to(device)
        labels = labels.to(device)

        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        running_corrects += torch.sum(preds == labels.data)

    with open('result.csv', 'w', newline='') as f:
        writer = csv.writer(f)
        writer.writerow(['y','pred'])
        for i in range(len(preds)):
            writer.writerow([str(labels.data[i].item()),str(preds[i].item())])

    epoch_acc = running_corrects.double() / len(dataloaders['test'].dataset)

    print('Testing completed')
    print('Acc: {:.4f}'.format(epoch_acc))
    return

def objective(trial):
    lr =trial.suggest_loguniform('lr', 1e-6, 1e-4)
    beta1 =trial.suggest_uniform('beta1', 0.8, 0.95)
    beta2 =trial.suggest_uniform('beta2', 0.9, 0.99)
    eps =trial.suggest_loguniform('eps', 1e-9,1e-7)

    model_ft = make_model('resnet18', num_classes = NUM_CLASSES, pretrained=True, input_size = (512,512))
    model_ft = model_ft.to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer_ft = optim.Adam(model_ft.parameters(),
    lr=lr, betas=(beta1, beta2), eps=eps, weight_decay=0, amsgrad=False)

    model_ft, best_acc, fig = train_model(model_ft, criterion, optimizer_ft, trial.number, num_epochs=NUM_EPOCHS)
    trial.set_user_attr(key="booster", value=model_ft)

    return best_acc

def callback(study, trial):
    if study.best_trial.number == trial.number:
        study.set_user_attr(key="best_booster", value=trial.user_attrs["booster"])
    return

data_dir = './'
BATCH_SIZE = 32
NUM_WORKER = 4
NUM_CLASSES = 2 # クラス数
NUM_EPOCHS = 10 # エポック
NUM_TRIALS = 10 # 試行数
GRAYSCALE = False # grayscale -> True

image_datasets = prepare_dataset(datasets.ImageFolder(os.path.join(data_dir, 'dataset')))
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val', 'test']}
dataloaders = create_dataloader(image_datasets)

study = optuna.create_study(direction='maximize')
finetuned_model = study.optimize(objective, n_trials=NUM_TRIALS, callbacks=[callback])
best_ft_model = study.user_attrs["best_booster"]
print('Testing the best model ...')
test_model(best_ft_model)
torch.save(best_ft_model.state_dict(), 'best_finetune_ResNet18.model')
```