広島県立総合技術研究所東部工業技術センター研究報告　No.37（2024）
技術ノート

深層学習による画像の領域分割（第２報）

PSPNet

廣川 勝久　花房 龍男　中濱 久雄

Image Segmentation using Deep Learning (Ⅱ)

PSPNet

HIROKAWA Katsuhisa, HANAFUSA Tatsuo and NAKAHAMA Hisao

大幅な性能向上が図られ、膨大な学習パラメータを持った深層学習モデルが毎年のように生み出されている。このような高性能な事前学習モデルをバックボーンとして実装した深層学習モデルが提案されている。本報告では、バックボーンに ResNet を実装した PSPNet [arXiv: 1612.01105, (2017)]の画像セグメンテーション能力の評価結果について述べる。実験により、画像データの水増し方法と最適化関数の違いから PSPNet の学習能力に差が生じることを示す。特に、アフィン変換と Mixup・Cutmix による水増し方法の PSPNet に対する有効性を明らかにする。

キーワード：データ拡張，PSPNet，転移学習，ResNet, Semantic Segmentation，葉の病気，leaf disease

1. 緒　　言

現在、深層学習による画像認識技術は人間の認識能力を凌駕するレベルに到達している。しかし、深層学習の性能向上を追求した結果、深層学習モデルに実装される学習パラメータ数は爆発的に増加している。これらの高性能モデルは高い学習能力を有するものの、その能力を十分に活用するには大量の教師画像と膨大な処理コストを必要とする。また、少量の教師画像を用いた場合、学習が不安定となり、過学習状態に陥りやすいという課題がある。

そのような状況において近年開発されたトップレベルの深層学習モデルでは、事前学習済みの基盤モデルが提供されることが一般的となっている。これらの事前学習モデルは、転移学習やファインチューニングを通じて、少量の教師画像と比較的少ない処理コストで他に転用することが可能となる点で注目されている。

これまで我々は、U-Net[1]モデルを用いた画像領域のセグメンテーション能力について評価を行い、その結果を報告した[2,3]。様々な画像が含まれる PASCAL VOC2012[4]の画像の学習では、U-Net は過学習の傾向を示すことが明らかとなった。本報告では、深層学習モデルに PSPNet[5]を使い、VOC2012 のデータを水増しすることにより、セグメンテーション能力向上を試みた。報告する PSPNet は学習済み ResNet[6]をバックボーンに転用したものである。

2. 転移学習と画像領域分割

2.1 学習済み ResNet バックボーン

PSPNet は、特徴抽出の機能部分を担うために学習済みの ResNet をバックボーンとして実装し、処理コストを抑えている。ResNet の実装により、PSPNet への入力画像は、画像に含まれる典型的な特徴量が２次元情報を保ちながら多次元の潜在変数として分類されるよう学習が行われる。

2.2 転移学習とファインチューニング

学習済み基盤モデルを用いる場合、転移学習とファインチューニングの 2 種類の学習方法から選択する。この選択には、学習すべきデータの量と基盤モデルの学習能力を比較する必要がある。一般的には、提供される教師データに対して基盤モデルの能力が高い場合は、転移学習を用い、教師データが多く転移学習では十分に学習出来ない場合は、ファインチューニングを行う。

ResNet を実装した PSPNet では、ResNet 以外の構造が学習パラーメータを持ち、加えて ResNet のパラメータの一部を変更し実装するため、変更されたパラメータも再学習されてしまう。このため、学習済みの ResNet 部分を転移学習とし学習パラメータを減少させた場合でも、PSPNet の学習能力は高いため、過学習状態に陥りやすい。

2024.4.15 加工技術研究部　受理

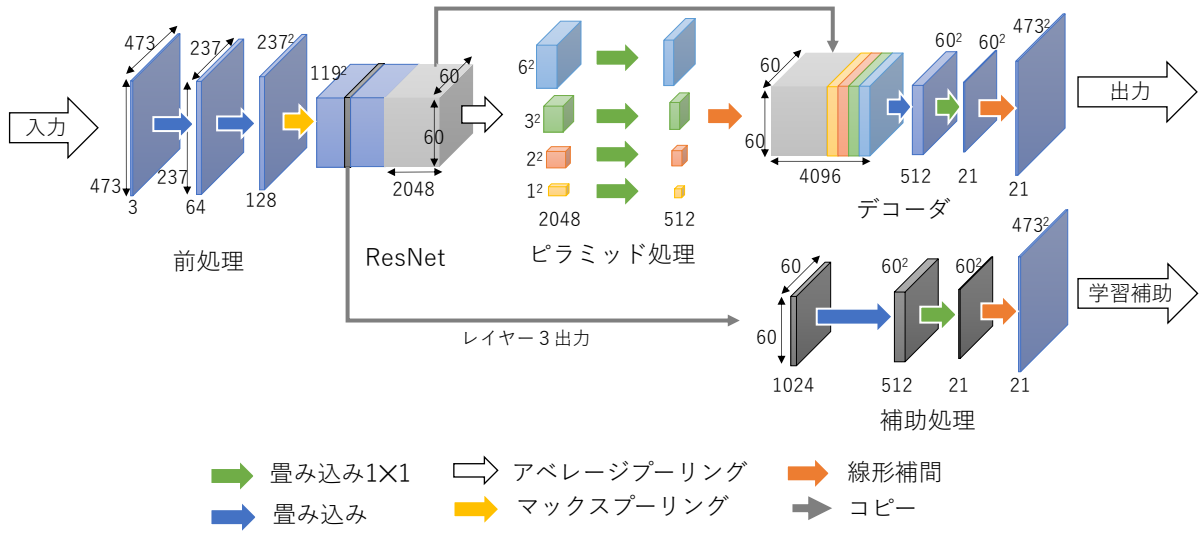

図 1　PSPNet の構造

過学習をさけるためには、PSPNet のさらなる学習パラメータ数の削減や教師画像の水増し（元画像の拡大や縮小、回転、位置の変更）学習率の調整などが必要となる。

2.3 PSPNet による画像領域分割

PSPNet は画像に含まれる様々な物体の領域を検出するための深層学習モデルである。**図 1** にその構造を示す。PSPNet は主に、前処理、ResNet、ピラミッド処理、デコーダ、補助処理の 5 つの部分から構成されている。

前処理部は畳み込みニューラルネットワークの構造を持ち、入力画像の局所的特徴を抽出しながら、入力画像の解像度を ResNet の入力解像度に一致させる。

ResNet はバックボーンとして機能する事前学習モデルである。但し、構造の一部が変更されているため、読み込んだ全ての事前学習パラメータを利用することは出来ない。ResNet では、多段の畳み込みニューラルネットワークとスキップ接続から構成され、入力画像の空間情報を保ちながら、最終的な特徴空間（60×60 のノード数から成る 2048 クラスタ）へと学習が行われる。

それに続く、ピラミッド処理が PSPNet の最も特徴的な構造と言える。U-Net では、プーリングと畳み込みと繰り返しながら、徐々に低解像度化された特徴値を学習により得ているが、PSPNet では、このピラミッド処理により ResNet が出力する特徴空間から 4 種類の異なる低解像度に対応した特徴値の並列な学習が行われている。4 種類の特徴値は ResNet の出力と同じ解像度に線形補間された後、ResNet からの特徴空間と結合される。

結合された特徴量は畳み込みニューラルネットワークによりデコードされ、入力画像と同じ解像度まで復元される。補助処理では、ResNet の中間層からの特徴値をスキップ接続する形で誤差学習の補助に利用している。

PSPNet を画像の領域分離に適用した場合、最終出力層の各クラスタが、領域分離の種類（ラベル）に対応する。**図 1** では 21 種類に領域を分けた例を示す。学習時の損失計算のために、まず最終出力層の各クラスタの同一座標の画素からソフトマックスの値を求める。次に得られたソフトマックス値と教師画像との交差エントロピーを損失関数として用いる。今、クラスタk面の画素座標Xが与えられた場合、ソフトマックス値は

$$p_k(X) = \frac{\exp\left(a_k(X)\right)}{\sum_{k'=1}^{K} \exp\left(a_{k'}(X)\right)} \tag{1}$$

と表せる。ここでKは最終出力層のクラスタ数を示す。出力に対応する領域を与える教師画像の画素座標Xのラベル値をlとすると、教師画像から PSPNet の出力クラスタ数と一致した教師画像クラスタ

$$q_k(X) = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases} \tag{2}$$

を生成する。これにより PSPNet からの出力と交差エントロピー関数は次式により求めることができる。

$$E = \sum_k \sum_X q_k(X) \log\left(p_k(X)\right) \tag{3}$$

補助処理の交差エントロピー計算も同様の計算により求める。

3. PSPNet の実装

3.1 画像データ

PSPNet の教師、評価用の画像として、Pytorch に準備されている画像セグメンテーション用 Datasets から VOCSegmentation を用いた。また、この画像データの読み込みには、Albumentations をフレームワークとして使

用している。これは、後述する画像データの水増し(データ拡張：Data Augmentation)に対応するためである。

3.2 データ拡張

まず、PSPNet の著者らが提案するデータ拡張には、ミラー反転、0.5〜2倍の画像のサイズ変更、±10°の回転、Gaussian blur（ぼかし）をランダムに適用した画像を用いており、以下のとおり Albumentations のフレームワークにより実装した。

```
augtransform = A.Compose([

        A.HorizontalFlip(p=0.5),
        A.ShiftScaleRotate(shift_limit=0, scale_limit=(-0.5,1.0), rotate_limit=10,
                           border_mode=0, p=1),
        A.GaussianBlur(always_apply=True),
        A.RandomResizedCrop (width=imagesize, height=imagesize, scale=(1.0, 1.0),
                             ratio=(1.0, 1.0)),
        A.Normalize(mean=(0), std=(1)),
        A.pytorch.ToTensorV2(transpose_mask=True),

])
```

Albumentation フレームワークでは、transpose_mask=True とすることにより、領域ラベル画像も教師画像のデータ拡張と同じように一致させることができる。

3.3 Cutmix と Mixup によるデータ拡張

本報告では、PSPNet のデータ拡張として Cutmix と Mixup の有効性についても確認を行った。Mixup は 2 枚の教師画像の各画素値を足し合わせて教師画像とする。今 i 枚目の教師画像を$T_i(X)$に j 枚目の教師画像$T_j(X)$を足し合わせる場合 Mixup の教師画像には次式に従い

$$T_i(X) = \alpha T_i(X) + (1-\alpha)T_j(X) \tag{4}$$

画像合成を行う。ここで、αは 0〜1 の範囲の乱数とする。

```
def mixup(data, target):
    batch_size = data.size(0)
    coe = torch.rand(batch_size)
    index = torch.randperm(batch_size)
    shuffled_data = data[index]
    shuffled_target = target[index]
    for i,alpha in enumerate(coe):
        data[i]   = alpha * data[i]+    (1 - alpha) * shuffled_data[i]
        target[i] = alpha * target[i] + (1 - alpha) * shuffled_target[i]
    return data, target
```

画像領域を示す画像についても(2)式の処理後、同様な処理を行う。Mixup では 2 枚の同じ座標の画素値を足し合わせたが、Cutmix では、2 枚の画像のエリアの足し合わせを行う。今 i 枚目の教師画像を$T_i(X)$とすると合成された教師画像は

$$T_i(X) = \beta T_j(X) + (1-\beta)T_i(X) \tag{5}$$

$$\because \beta = \begin{cases} 1 & \text{if } w_1 \le w < w_2,\ h_1 \le h < h_2 \\ 0 & \text{otherwise} \end{cases}$$

と表せる。ここで、w, hは教師画像$T_j(X)$から切り出されるランダムな大きさの画像エリアを示す。

```
def cutmix(data, target):
    batch_size = data.size(0)
    width      = data.size(2)
    height     = data.size(3)
    indices    = torch.randperm(batch_size)
    shuffled_data   = data[indices]
    shuffled_target = target[indices]
    coe = torch.rand(batch_size)                                    #0~1の乱数
    for i,beta in enumerate(coe):
        cut_width  = int(width  * (1 - beta))
        cut_height = int(height * (1 - beta))
        cx = np.random.randint(width)
        cy = np.random.randint(height)
        bbx1 = np.clip(cx - cut_width  // 2, 0, width )
        bby1 = np.clip(cy - cut_height // 2, 0, height)
        bbx2 = np.clip(cx + cut_width  // 2, 0, width )
        bby2 = np.clip(cy + cut_height // 2, 0, height)
        data  [i, :, bby1:bby2, bbx1:bbx2]  = shuffled_data  [i, :, bby1:bby2, bbx1:bbx2]
        target[i, :, bby1:bby2, bbx1:bbx2]  = shuffled_target[i, :, bby1:bby2, bbx1:bbx2]
    return data, target
```

実装にあたっては、バッチ処理ごとに Mixup と Cutmix をランダムに選択する学習とした。

3.4 PSPNet の実装

PSPNet を実装するためには、ResNet をバックボーンとした転移学習モデルを組み込む必要がある。本稿では、ResNet50 を実装した。まず、ResNet50 の学習済みのパラメータを ResNet50 に読み込み、勾配計算を行わないように設定を行うことで、ResNet 部分が転移学習となる。但し、ResNet の構造に変化を加えた場合は、その部分の再学習が行われる。

```
    weights = ResNet50_Weights.DEFAULT
    resnet = resnet50(weights=weights)
    for param in resnet.parameters():
        param.requires_grad = False
    self.input = InputLayer(classter1, classter2, classter3)

    self.layer1 = resnet.layer1
    self.layer2 = resnet.layer2
    self.layer3 = resnet.layer3
    self.layer4 = resnet.layer4
```

PSPNet の構造は非常に簡単なプログラムにより記述できる。入力画像の解像度に合わせた入力用 2 次元畳み込みニューラルネットワークからの出力を ResNet50 に接続後、ResNet50 の 3 番目のレイヤー出力を補助処理として処理途中に取り出す。また、ResNet50 を通過した出力はピラミッド処理、デコーダーへと接続する。

```
def forward(self, img):
    img = self.input(img)
    img = self.layer1(img)
    img = self.layer2(img)
    img = self.layer3(img)
    aux = self.aux(img)
    img = self.layer4(img)
    img = self.pyramid(img)
    img = self.decoder(img)
    return img, aux
```

PSPNet では、ResNet50 のレイヤー 1 の入力層に対して以下の変更が加えられている。

```
self.layer1[0].conv1 = nn.Conv2d(
    classter3, classter2, kernel_size=(1, 1), stride=(1, 1), bias=False)
self.layer1[0].downsample[0] = nn.Conv2d(
    classter3, classter4, kernel_size=(1, 1), stride=(1, 1), bias=False)
```

ピラミッド処理では、ResNet50 から出力される 60 画素×60 画素のデータがアベレージプーリングにより低解像度化され、2 次元畳み込みニューラルネットワークに

より、クラスターの圧縮が行われる。

```
        self.pool = nn.AdaptiveAvgPool2d(size)
        self.conv = nn.Conv2d(L1, L2, kernel_size=1, bias=False)
        self.bn = nn.BatchNorm2d(L2)
        self.relu = nn.ReLU()

    def forward(self, img):
        x = self.pool(img)
        x = self.conv(x)
        x = self.bn(x)
        x = self.relu(x)
        return x
```

このピラミッド処理により 4 種類の解像度それぞれに対応した特徴値に低次元化される。低次元化された特徴値は再び線形補間により、解像度を ResNet の解像度に戻された後 ResNet からの出力と合成され、デコーダの入力と成る。

```
        self.p1 = Pyramid_pooling(classter, out_c_size, pyramid_size[0])
        self.p2 = Pyramid_pooling(classter, out_c_size, pyramid_size[1])
        self.p3 = Pyramid_pooling(classter, out_c_size, pyramid_size[2])
        self.p4 = Pyramid_pooling(classter, out_c_size, pyramid_size[3])

    def forward(self, img):
        size = img.size()
        x1 = self.p1(img)
        x1 = F.interpolate(x1, size=size[2:], mode="bilinear", align_corners=True)
        x2 = self.p2(img)
        x2 = F.interpolate(x2, size=size[2:], mode="bilinear", align_corners=True)
        x3 = self.p3(img)
        x3 = F.interpolate(x3, size=size[2:], mode="bilinear", align_corners=True)
        x4 = self.p4(img)
        x4 = F.interpolate(x4, size=size[2:], mode="bilinear", align_corners=True)
        img = torch.cat([img, x1, x2, x3, x4], dim=1)
        return img
```

学習時の誤差は、PSPNet から出力される領域ラベル画像と教師領域ラベル画像の交差エントロピーに補助処理と教師領域ラベル画像の交差エントロピーを加えたものとした。

```
        model.zero_grad()
        outputs, auxloss = model(images)
        loss = criterion(outputs, targets) + alpha * criterion(auxloss, targets)
        loss.backward()
        optimizer.step()
```

最適化関数には SGD を実装し、学習係数 0.01 に重みの忘却係数 0.0001 が追加されている。また、学習が進むにつれ学習係数が、次式のように減衰する設定とした[4]。

$$lr = \left(1 - \frac{iter}{maxiter}\right)^{power} \tag{6}$$

実験では power=0.9 を選択した。

```
optimizer = optim.SGD(model.parameters(), lr=0.01,weight_decay=0.0001)
```

```
lambda1 = lambda epoch : (1-epoch/EPOCHS)**0.9
scheduler = optim.lr_scheduler.LambdaLR(optimizer,lambda1)
```

実験では、最適化関数 Adam との比較によりデータ拡張の効果の違いについても検討した。

```
optimizer = optim.Adam(model.parameters(), lr=0.01, betas=[0.9, 0.999],
                       weight_decay=0.0001)
```

4. PSPNet の評価

4.1 反転・サイズ変更・回転・ぼかしによるデータ拡張画像

データ拡張を PSPNet に適用するため、データ拡張が適切に行われているかを確認した。まず VOCSegmentation 画像に対してミラー反転、サイズ変更、回転、Gaussian blur を行った画像を**図 2** に示す。Albumentations フレームワークを用いたことにより、領域ラベル画像にも入力画像と同じように反転、サイズ変更、回転の画像処理が行われ領域ラベル画像と入力画像の位置に差は生じていない。学習時には、エポック毎に VOCSegmentation 画像にデータ拡張の再処理を行った学習画像を用いた。

4.2 Mixup と Cutmix によるデータ拡張画像

図 3 には、Mixup によるデータ拡張画像を示す。入力画像からは、2 枚の画像の重なり合いが見て取れる。これは、2 枚の画像の比率を変えながら足し合わせた効果である。一方、領域ラベル画像は、座標ごとに最大値を持つクラスターをラベル画像化しているため、重なりは確認出来ない。Cutmix では、サイズの異なる矩形がランダムな位置に配置されているのが教師画像、領域ラベル画像共に確認できる（**図 4**）。

4.3 PSPNet の学習能力評価

PSPNet の評価のために、VOCSegmentation から学習用画像 1400 枚と評価用未学習画像 500 枚をランダムに使い、学習時のエポック毎の誤差を計算した。**図 5** にその学習曲線を示す。実線が教師画像に対する学習誤差を表し、破線は未学習画像に対する誤差を表す。データ拡張を行わない画像（緑線）については、教師画像は学習されるが、未学習画像については誤差が大きな状態に留まる。ミラー反転、サイズ変更、回転、ぼかしを行った画像に対する学習誤差（青線）とデータ拡張を行わない場合とを比較すると、教師画像、未学習画像共に学習誤差が低減されている。Mixup・Cutmix を組み合わせたデータ拡張では、未学習画像について最も学習誤差を低減することが出来た（赤線）。但し、学習が進むにつれ、未学習画像に対する学習誤差が僅かに増加する傾向が見られた。

次に教師画像と未学習画像からそれぞれランダムに 100 枚を選び、Mixup・Cutmix のデータ拡張による学習後の PSPNet に入力しその画像領域の分割能力について画像により確認を行った。**図 6〜図 9** に誤差の値順に並べた結果を示す。**図 6**、**図 7** は教師画像に対して誤差が小さな画像と大きな画像をそれぞれ 16 枚示す。画像中の領域のサイズが小さく形が単純な方が誤差は小さい傾向にある。逆に複雑な画像は誤差が大きくなる傾向が見られる。この傾向は**図 8**、**図 9** の未学習画像についても同様ではあるが、誤差が大きな教師画像と未学習画像同士を比較した場合、未学習画像の方が画像の領域分割が不十分であ

った。ただしデータ拡張を行わない U-Net の学習と比較すると、大幅な能力の向上が見られた。

最後にデータ拡張の手法に対する最適化関数について報告する。**図 5** のとおり最適化関数 SDG では、Mixup・Cutmix のデータ拡張が最も誤差が少なく、ミラー反転、サイズ変更、回転、ぼかしのデータ拡張と性能差が生じた。最適化関数を SDG から Adam に変更した場合の学習曲線を**図 10** に示す。最適化関数が Adam の場合、両方のデータ拡張手法ともほぼ同程度の学習能力となった。また、最適化関数が SDG の場合では Mixup・Cutmix のデータ拡張に、僅かに過学習の傾向がみられたが、Adam ではその傾向も解消されている。

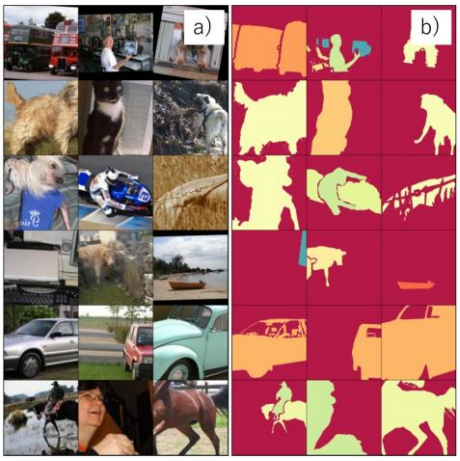

図 2　反転・サイズ変更・回転・
ぼかしによるデータ拡張
a) 入力画像，b) 領域ラベル画像

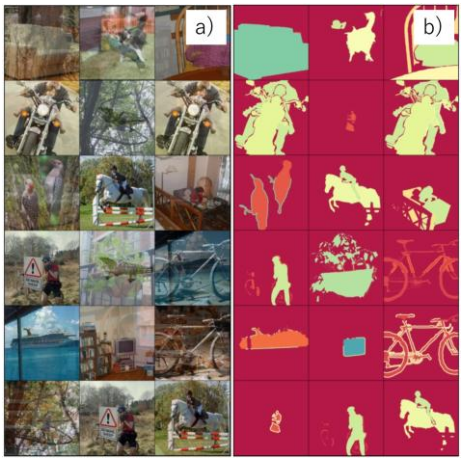

図 3　Mixup データ拡張
a) 入力画像，b) 領域ラベル画像

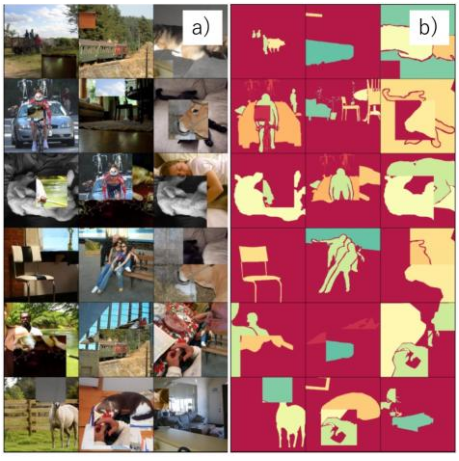

図 4　Cutmix データ拡張
a) 入力画像，b) 領域ラベル画像

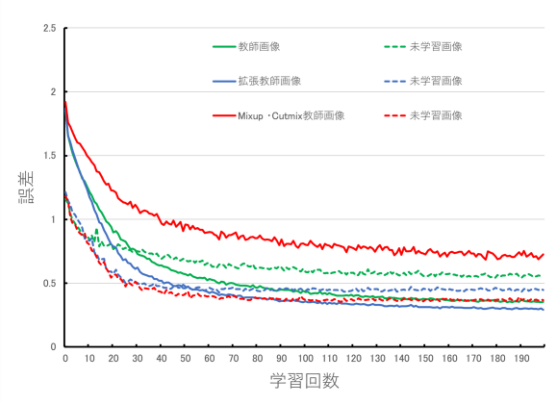

図 5　データ拡張と学習誤差

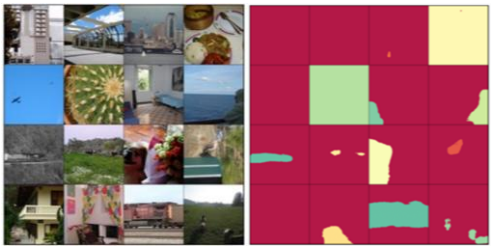

図 6　教師画像（誤差少）

図 7　教師画像（誤差大）

図 8　未学習画像（誤差少）

図 9　未学習画像（誤差大）

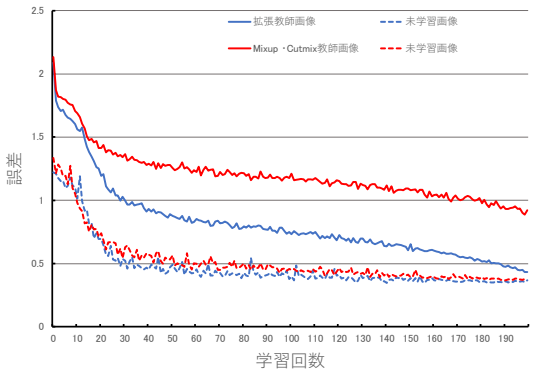

図 10　データ拡張と学習誤差

4.4 葉の病気領域検出に対する評価

VOCSegmentation では、21 種のクラスへの領域分割能力評価を行ったが、出力を 1 クラスとすることにより、領域をアナログ値出力することも可能となる。実験では、植物の葉の一部が病気に侵された領域データ [7] を学習させ、未学習の葉画像の病気領域検出能力について評価した。**図 11** に未学習の入力画像と**図 12** には未学習画像に対応した病気領域画像を示す。入力画像に対して PSPNet が検出した病気領域をアナログ出力した画像を**図 13** に示す。全般的にアナログ値出力とすることで、病気領域の確からしさが出力されている。これは、健全な領域と病気の領域がはっきりと区別がつかない領域については濃度を下げた画素として表示することで、人間の判断と同じようにあいまいな表現を可能とすると考えられる。また、それぞれの入力画像を個別に評価した場合、**図 13** の青丸は、ラベル画像には無い人間が見落とした病気領域

図 11　未学習画像（病気の葉）

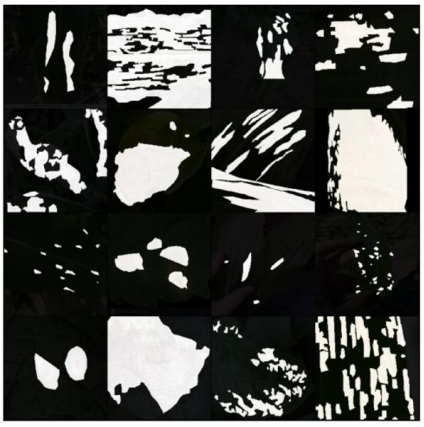

図 12　領域ラベル画像

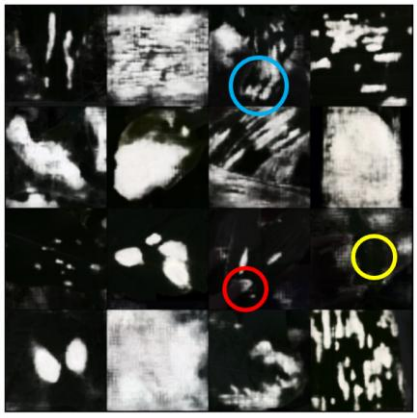

図 13　領域画像出力

を検出している。一方赤丸は人間の手を病気領域として検出し、黄色の丸は病気の領域が検出されないなどの誤認識も確認された。

5. 結　　言

バックボーンに ResNet が実装された PSPNet の画像のセグメンテーション能力について評価した。ResNet 部分

を学習が行われない転移学習に設定した PSPNet の学習においても、高い学習能力を示し、PASCAL VOC2012 の 1400 枚の画像だけでは、学習には不十分であった。そのため、ミラー反転、サイズ変更、回転、ぼかしの処理を教師画像に行う方法と Mixup・Cutmix をランダムに行う方法により学習データ拡張した学習を行った。その結果、両方の方法とも学習誤差が低減され、有効性を確認した。

学習パラメータが非常に多い高性能なモデルの学習には、大手 IT 企業のような膨大な学習データの収集能力と、大型コンピュータのような処理能力が必要な場合も増加している。そのため、高性能な基盤モデルをバックボーンとして実装することは、計算コストを抑えて高性能な深層学習モデルの応用開発には、今後重要な技術になり得ると考えられる。

文　　献

1） Olaf Ronneberger, Philipp Fischer, and Thomas Brox,: U-net: Convolutional networks for biomedical image segmentation, arXiv: **1505.04597**, (2015).

2） 廣川 勝久，花房 龍男，中濱 久雄：広島県立総合技術研究所東部工業技術センター研究報告，深層学習による画像の領域分割（第１報）**36**（2023）.

3） 花房龍男, 廣川勝久, 中濱久雄：広島県立総合技術研究所東部工業技術センター研究報告，深層学習を用いた溶射皮膜の空孔率算出 U-NET による溶射皮膜空孔部の領域分割，**37**（2024）.

4） WWW：Visual Object Classes Challenge 2012 (VOC2012)，http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html.

5） Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, Jiaya Jia: Pyramid Scene Parsing Network, arXiv: **1612.01105**, (2017).

6） Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: Deep residual learning for image recognition, arXiv:**1512.03385** (2015).

7） WWW: Leaf disease segmentation dataset, https://www.kaggle.com/datasets/fakhrealam9537/leaf-disease-segmentation-dataset/.