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.36（2023）
技術ノート

深層学習による画像の領域分割（第１報）

ユーネット

廣川　勝久　花房　龍男　中濱　久雄

Image Segmentation using Deep Learning (Ⅰ)

U-Net

HIROKAWA Katsuhisa, HANAFUSA Tatsuo and NAKAHAMA Hisao

近年，スマートフォン，自動車，監視カメラなどから撮影された画像が様々な産業や分野で利用されている。撮影された非常に多くの画像から必要とする情報を得るための方法として，深層学習の技術が用いられようとしている。本報告では，深層学習モデルであるU-Net[arXiv:1505.04597,(2015)]の実装を行い，その画像領域の分割能力について評価を行った。評価には，一般的な画像とドローンから撮影した画像を用いて行い，両者の学習の違いなどについて述べた。また，U-Net 実用化のために，重みの学習とその利用方法について説明を行った。

キーワード：ユーネット，セグメンテーション，画像，U-Net，Neural Network，

1. 緒　　言

畳み込みニューラルネットワークの原型であるネオコグニトロン [1)]が開発され，ニューラルネットワークは画像処理へと応用されるようになり，ここ 10 年間の画像処理用ニューラルネットワークの進歩には著しいものがある。これには，スキップ接続や，ReLU 関数などの新しいテクニックが生み出されたことによる。これらのテクニックにより，これまで，不可能とされた数十層以上にも多層化されたニューラルネットワークを学習させることが可能となった（深層学習）。深層化されたニューラルネットワークは，これまでより複雑で，多くの情報を学習でき，人間のハンドコーディングによる画像処理方法を遥かに凌駕する能力を学習により獲得するようになった。

我々はこれまで，生成モデルの一種であるオートエンコーダ（**A**uto **E**ncoder: **AE**）の高い汎化能力による異常検知について報告した [2)]。また，生物の視覚情報処理に類似した畳み込みニューラルネットワークをオートエンコーダに実装した 2 次元ニューラルネットワークのクラスタリング能力の異常検知能力についても報告を行った [3)]。本稿では，これらニューラルネットワーク構造を基礎に持つU-Net[4)]の実装を行い，その能力評価の結果について報告する。U-Net は画像処理のために開発された深層学習モデルであり，画像から必要とする物体の領域を検出する能力を持つ。

2. 2 次元畳み込みによる画像領域分割

2.1 オートエンコーダによる低次元化と特徴値

オートエンコーダ（**A**uto **E**ncoder: **AE**）は深層学習による生成モデルの基盤となる技術の一つである。オートエンコーダでは，まず，多次元の学習データを入力とし，次元数を減らしながら，各データに共通する典型的な特徴部分を低次元の潜在変数へと学習する。学習後には，潜在変数の座標から，対応する典型的なデータを生成することが可能となる。学習後，学習データを入力し，潜在変数の座標を調べてみると，学習データに応じた典型的な特徴により分類が行われていることを確認できる。この機能は，本稿のニューラルネットワークの構造にも応用されている。学習データの持つ典型的な特徴画像（模様，色，形など）を低次元化した潜在変数に学習させることにより，画像の特徴に応じた分類が行われる。

2.2 畳み込みニューラルネットワーク

従来型のニューラルネットワークが全結合型の重みで構成されるのに対して，受容野と呼ばれる局所結合型の重みを持つものに畳み込みニューラルネットワークがある。畳み込みニューラルネットワークは，画像処理分野へ広く応用される重要な技術である。これは，カメラなどで撮影された画像には，処理の対象となる物体が必ずしも同じ位置には存在しないが，異なった位置に存在する対象物に対しても同じ処理を必要とするからである。畳み込みニューラルネットワークは局所結合の重みにより，対象物の位置によらない処理能力を実現している。この畳み込みニューラルネットワークにオートエンコーダを組み合わせることで位置情報を保ちながら，入力データの典型的な特徴量を低次元の潜在変数へと学習を行うことが出来る。

2.3 U-Net による画像領域分割

画像に含まれる様々な物体の領域を検出する場合，①物体の持つ特徴量を学習する。②特徴量に応じた物体の位置と形状を検出する。と言った２つの機能が必要となる。これは，前述の畳み込みニューラルネットワークとオートエンコーダの組み合わせによりある程度は可能ではあるが，潜在変数への低次元化を行うエンコード時に位置

2023.8.21　加工技術研究部　受理

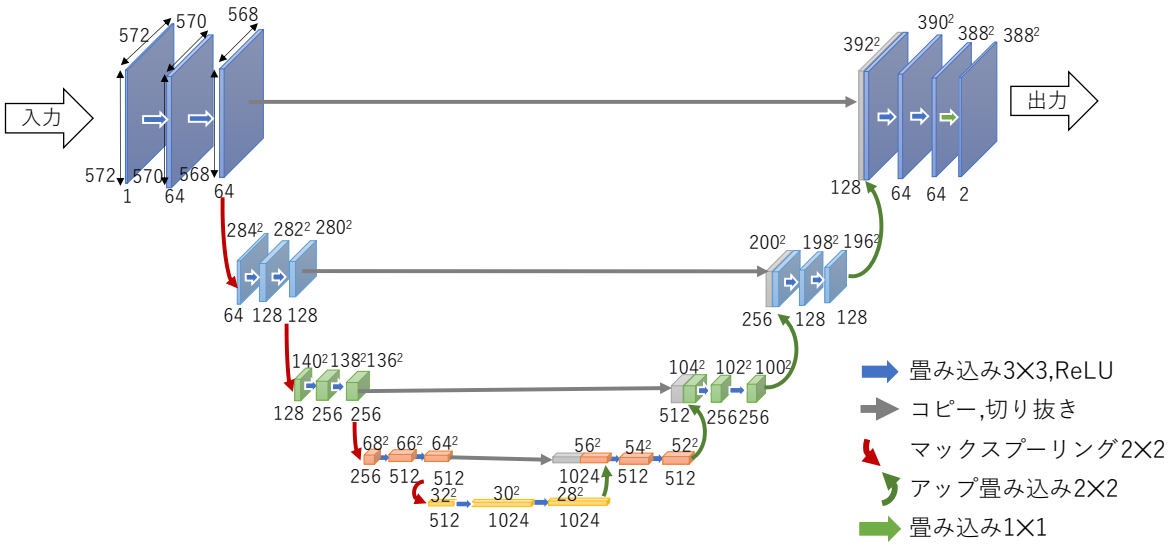


図 1　U-Net の構造

情報も同時に減少するため，デコードされた画像から，入力データと同等の形状を得ることは出来ない。U-Net はこの課題を克服した深層学習モデルである。**図 1** にその構造を示す。画像を対象とするため U-Net には全結合の重みは使用されておらず，全ての層が畳み込みにより結合された特徴を持つ。まず，入力画像は，２層の畳み込み層を通過すると，1/4 の解像度に圧縮後，次の処理に送られ同様の処理が行われる。これを，繰り返しながら，画像に含まれる様々な物体の特徴量を潜在変数へと抽出・学習して行く。潜在変数からは，これまでとは逆に４倍の解像度に拡大され，２層の畳み込み層を通過しながら，同様の処理を繰り返し行うことで，最終的に入力画像と同等な解像度となるまで処理が繰り返される。U-Net では同じ解像度のエンコーダとデコーダとの層をスキップ接続することによって，解像度の回復を図っている。この処理では，デコーダからの特徴量とエンコーダからスキップ接続した高解像度の情報から，特徴量の位置を決定する。なお，エンコード時には画像の圧縮に対して，クラスタ数を増加させることで情報量の急激な低下が発生しない構造となっている。

U-Net の学習では，最終出力層の各クラスタが，分離すべき物体の種類（ラベル）に対応する。**図 1** では２種類の分離クラスタの場合を表す。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する。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

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

と表せる。ここでKは最終出力層のクラスタ数を示す。交差エントロピーの損失関数は画素ごとに次式により計算される。

$$E = \sum_{X} w(X) \log\left(p_{l(X)}(X)\right) \qquad (2)$$

lは各画素の教師画像ラベルを，$w(\cdot)$は学習時に重要な画素に対する重みを表す。

3. U-Net の実装

3.1 画像データ

U-Net を実装するに当たり，評価用の画像として PASCAL VOC2012 を用いた [5]。VOC2012 から提供された画像は縦横の画素数が統一されておらず，また，画像の分割領域を表すラベル画像がないものが多数ある。そのため，領域分割に必要とするラベル画像が存在する 2913 枚の画像を選んだ。これらの画像とラベル画像を正方形に切り出し，U-Net の入力層の解像度に一致するようにサイズ変更を行った。原著論文では，U-Net の解像度は 572×572 画素ではあるが，本稿では，1/4 の大きさにサイズ変更し，286×286 画素を用いて実装を行った。これは，572×572 画素では PC のリソースを多く消費し，長時間の計算を必要とするためである。

3.2 U-Net の実装

解像度が異なる各層の２回の畳み込みでは，出力に ReLU 非線形関数を用いた。

```
class Convolution(nn.Module):
    def __init__(self, L1, L2):
        super().__init__()
        self.conv1 = nn.Conv2d(L1, L2, kernel_size=3)
        self.conv2 = nn.Conv2d(L2, L2, kernel_size=3)
        self.bn = nn.BatchNorm2d(L2)
        self.relu = nn.ReLU()

    def forward(self, img):
        x = self.conv1(img)
        x = self.relu(self.bn(x))
        x = self.conv2(x)
        x = self.relu(x)
        return x
```

畳み込み層の前層のみ,BatchNorm2dによる重みの正規化を行った。この 2 層の畳み込み層の出力をプーリング層に入力し，エンコーダ層からの出力としている。

```
class DownConv(nn.Module):
    def __init__(self, L1, L2):
        super().__init__()
        self.conv = Convolution(L1, L2)
        self.pool = nn.MaxPool2d(2)

    def forward(self, img):
        large = self.conv(img)
        small = self.pool(large)
        return large, small
```

各エンコーダ層からは，入力同サイズの画像と 1/4 に圧縮された画像が出力される構造とした。次にデコーダ部分の層を示す。

```
class UpConv(nn.Module):
    def __init__(self, L1, L2, L3, flag):
        super().__init__()
        self.conv = Convolution(L1, L2)
        if flag == 0:
            self.uconv = nn.ConvTranspose2d(L2, L3, kernel_size=2, stride=2)
        else:
            self.uconv = nn.ConvTranspose2d(L2, L3, kernel_size=1, stride=1)

    def forward(self, x, y, size):
        if size != 0:
            y = trim(y, size)
            x = torch.cat([x, y], dim=1)
        x = self.conv(x)
        x = self.uconv(x)
        return x
```

デコーダ層では，スキップ接続により入力される画像と低解像度の層から高解像度化した画像を同一層内のクラスタとし,エンコーダ部と同様に2層の畳み込みを行う。その後，高解像度化のための畳み込みを行う。この畳み込みでは，出力に ReLU 非線形関数は用いていない。これらのエンコーダ部とデコーダ部を次のように接続しU-Netの実装を行った。

```
        self.layer1 = DownConv(classter1, classter2)
        self.layer2 = DownConv(classter2, classter3)
        self.layer3 = DownConv(classter3, classter4)
        self.layer4 = DownConv(classter4, classter5)
        self.layer5 = UpConv(classter5, classter6, classter7, flag=0)
        self.layer6 = UpConv(classter6, classter7, classter8, flag=0)
        self.layer7 = UpConv(classter7, classter8, classter9, flag=0)
        self.layer8 = UpConv(classter8, classter9, classter10, flag=0)
        self.layer9 = UpConv(classter9, classter10, channels, flag=1)

    def forward(self, img):
        org1, down1 = self.layer1(img)
        org2, down2 = self.layer2(down1)
        org3, down3 = self.layer3(down2)
        org4, down4 = self.layer4(down3)
        up5 = self.layer5(down4, _, 0)
        up6 = self.layer6(up5, org4, Layer6size)
        up7 = self.layer7(up6, org3, Layer7size)
        up8 = self.layer8(up7, org2, Layer8size)
        img = self.layer9(up8, org1, Layer9size)
        return img
```

最終層のみ，高解像度化のための畳み込みは行わず，畳み込み層の各クラスタの同画素位置から接続重みとした。

最適化関数には Adam を，損失関数には Cross-EntropyLoss()を用いた。実装・評価に当たり，(2)式の $w(X)$による画素ごとの重み付けは行っていない。

4. U-Net の評価

4.1 一般的な画像に対する画像領域分割の能力評価

VOC2012 の画像 2913 枚による U-Net の評価に当たり，2913 枚の内，2500 枚の画像を用いて学習し，残りの 413 枚を未学習の評価用画像とした。VOC2012 では，RGB の 3 種類の入力クラスタに対して，人や航空機など 21 種類がラベル付けされている。**図 2** に学習回数に対する損失誤差の値を示す。VOC2012 画像の学習では，学習が進むにつれ学習画像に対する誤差は減少するものの，未学習画像については，誤差の低下が見られない。

図 3，**図 4** は 1000 回の学習後，学習に用いた画像から画像 100 枚をランダムに選び，U-Net による画像の領域分割を行ったものを示す。**図 3** は損失誤差が小さい 25 枚の入力画像に対する領域分割の画像である。**図 4** には損失誤差の大きなもの 25 枚を示す。図中では,それぞれ(a)入力画像,(b)ラベル画像,(c)出力画像を表す。**図 3**，**図 4** から画像領域がほぼ一種類のラベルであれば，誤差は小さく，形状が複雑な場合や画像の占める領域が小さい場合には，損失誤差が大きくなる傾向がある。また，**図 5** は 1000 回の学習後，同様に未学習の画像に対する領域分割の結果を示す。損失誤差の小さな 25 枚の画像を選んだ。未学習画像では，全く画像と分割領域が一致しない結果となった。

そこで，損失誤差に対する画像の枚数についてグラフ化を行った。**図 6** に，学習を 400 回，700 回，1000 回行った U-Net の損失誤差を示す。教師画像については，学習が進むにつれ，損失誤差の大きな画像が減少する。一方未学習画像については，学習の回数によらない分布となった。

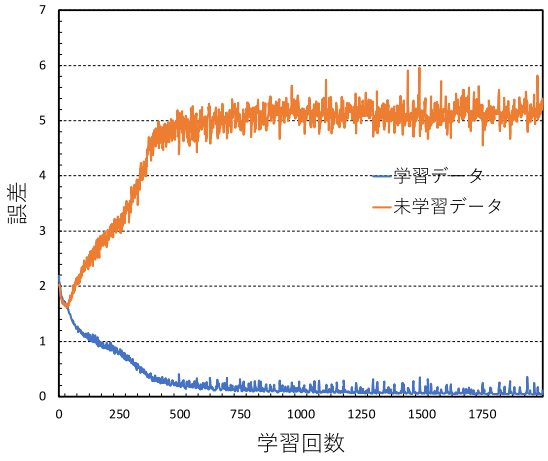
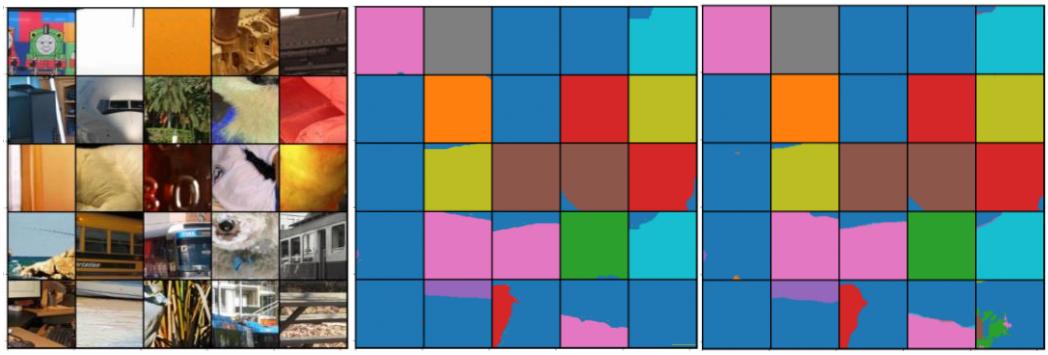
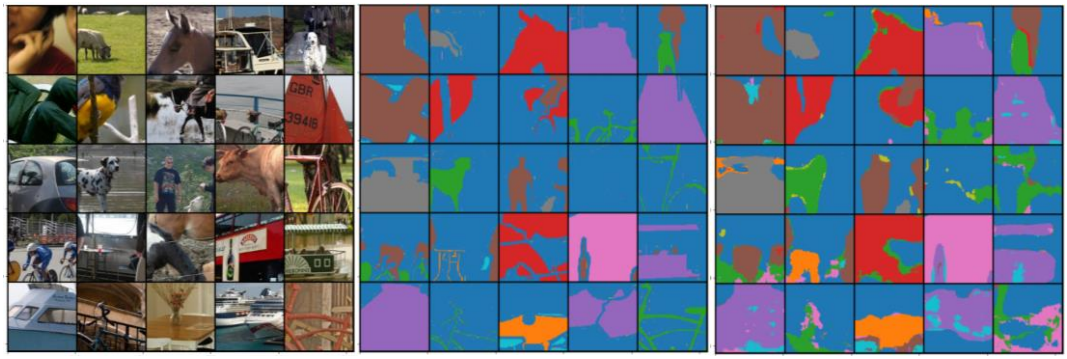


図 2　U-Net の学習曲線



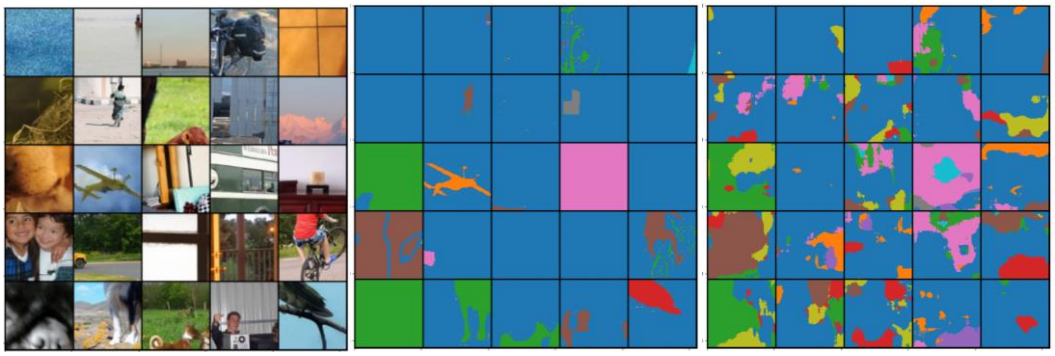
(a)　　(b)　　(c)

図 3　学習した画像に対するラベリング：損失誤差（小）
a) 入力画像, b) ラベル画像, c ） 出力画像



(a)　　(b)　　(c)

図 4　学習した画像に対するラベリング：損失誤差（大）
a) 入力画像, b) ラベル画像, c ） 出力画像



(a)　　(b)　　(c)

図 5　未学習画像に対するラベリング, a) 入力画像, b) ラベル画像, c ） 出力画像

本実験では，1/4 の大きさにサイズ変更した 286×286 入力画素に加えて，クラスター数も 1/4 に減少させ，学習パラメータ数が減少したモデルを用いた。しかし，VOC2012 の画像に対する**図 2** などの実験結果は，過学習状態の可能性を示していることから，過学習回避には教師データの更なる補充と適正な学習パラメータの設定が必要と思われる。

4.2 ドローン画像を用いた能力評価

U-Net はシフトインバリアントな構造を持つことから，ドローンにより撮影した画像から部分的に切り出した画像を教師画像として学習させた後，ドローンからの全体画像を入力した場合の領域分割について評価した。ドローン画像には，スイスと奥多摩の地域を撮影した画像を用いた [6]。これらの画像には，それぞれ 4608×3456 画素の 100 枚，3840×2160 画素の 91 枚が提供されている。ラベルには人や建物など 10 種類が設定されている。学習のための事前準備として，191 枚の画像から，2500 枚の

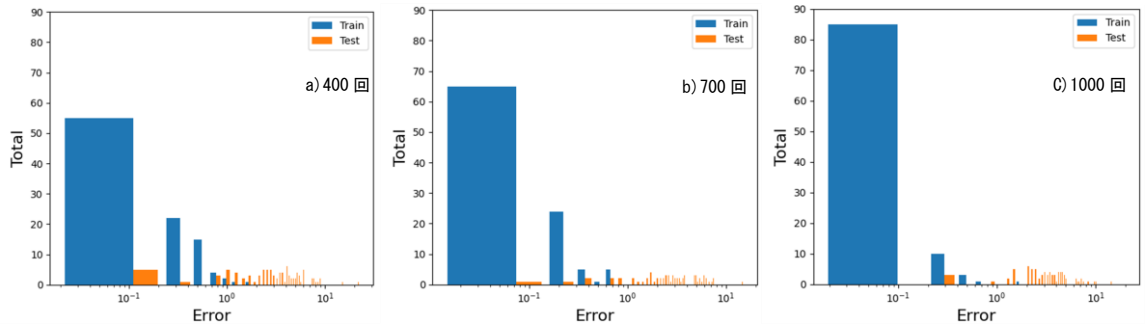
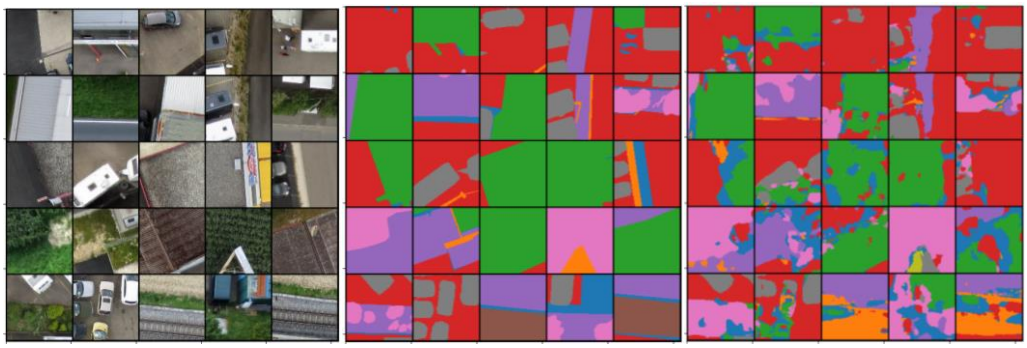
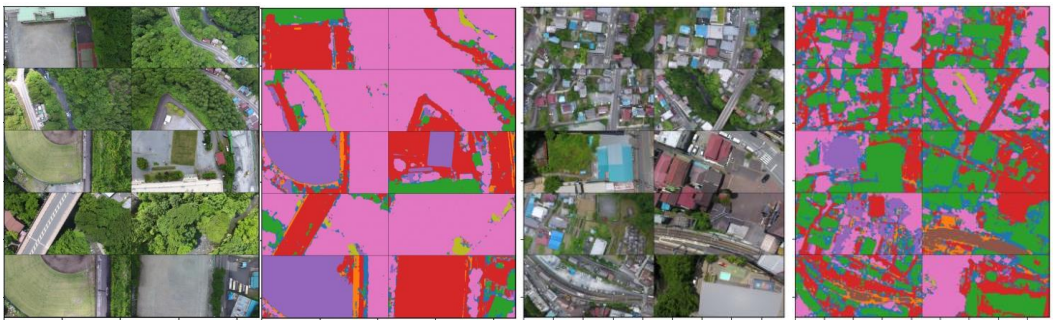


図 6　学習回数に対する損失誤差



(a)　(b)　(c)

図 7　未学習画像に対するラベリング, a) 入力画像, b) ラベル画像, c) 出力画像



(a)　(b)　(c)　(d)

損失誤差(小)　損失誤差(大)

図 8　ドローン画像による画像領域分割,
a) 入力画像, b) a) に対する出力画像, c) 入力画像, d) c) に対する出力画像

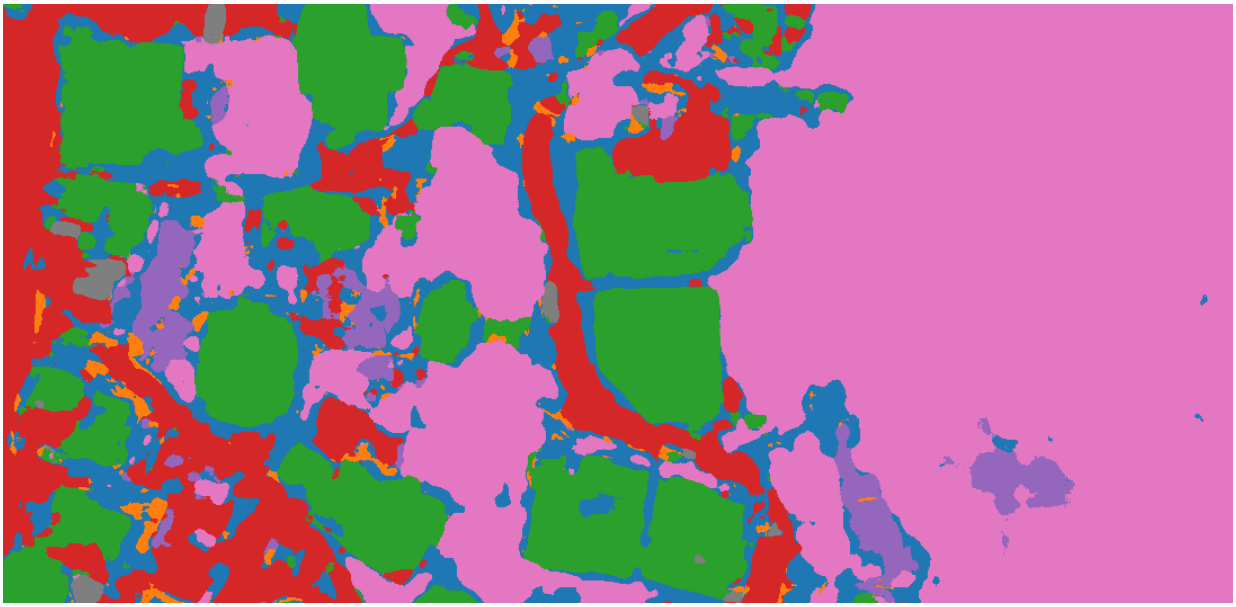
学習画像と，296 枚の未学習画像を切り出した。まず，高解像度の画像のランダムな位置から，858×858 画素を切り出し，286×286 画素まで画像を圧縮した。ラベル画像についても同じ位置から切り出し，圧縮を行った。

図 7 は学習後に，未学習の画像の入力対して，損失誤差の最も大きな 25 枚を表す。ドローン画像では，一般画像と異なり，未学習の画像についても領域分割が行われている。これは，未学習画像と学習した教師画像がよく似ているためと考えられる。

最後に, 学習した重みを用いてドローン画像全体の画像領域分割を行った。画像の大きさが変化した場合でも，U-Net では，入力画像のサイズのみを変更することで, 学習後の重みを読み込むことで再利用可能である。1/3 に圧縮した画像を用いて U-Net の学習を行ったため，まずドローン画像の解像度を 1/3 に圧縮した。次に全ての入力画像のサイズを統一するため，1280×720 画素に切り出した。

図 8 には奥多摩の画像を入力した場合の損失誤差の小さい 10 枚(a, b) と損失誤差の大きな 10 枚(c, d) を示す。画像の一部を学習した U-Net には，未学習の画像領域が存在すると思われるが, 全体の画像を入力した場合でも，領域分割は比較的正確に行われた。また, 単純な画像と，

(a)



(b)

図 9　詳細な領域分割画像:a)入力画像，ｂ）出力画像

複雑な画像を比較すると，この場合も複雑な場合の方が損失誤差は大きい。また，評価画像に未学習のスイス画像も含まれていたが，未学習のスイス画像は奥多摩画像と比較して損失誤差は小さかった。これは，スイス画像が非常に単純な画像から構成されていることが要因であると考えられる。

詳細な画像領域分割を確認するために，損失誤差が奥多摩画像の中で中間の値となる画像を**図 9** に示す。森や畑，道路や建物の分割が行われ，小さな自動車についても認識されているのが分かる。以上のドーン画像を用いた検証結果から，U-Net では，学習は画素数の小さな教師画像で行い，学習重みを実際の高解像画像に適用することが可能であり，計算リソースを有効に利用できる。すなわち，学習には，画像領域分割を行う全サイズの画像を用いる必要は無く，切り出した小さなサイズの画像から学習を行うことにより，計算能力の低い CPU や GPU を用いることが可能である。

5. 結　　言

画像に含まれる物体の領域を分割する手法として U-Net を実装し，その評価を行った。U-Net の実装おいて，原著

論文と同じ解像度画像を入力とした場合，多くの PC のリソースを必要とするため，1/4 倍まで解像度を下げて評価した。VOC2012 に含まれる一般的な画像では，学習に用いた画像の画像領域分割は学習が進むにつれ，損失誤差が減少した。しかし，未学習の画像については，損失誤差の減少は見られない過学習の状態となり，画像領域分割は的確に行われなかった。一方ドローン画像の画像領域分割は未学習画像についても，適用することが出来た。これは，ドローン画像では，未学習画像と学習画像の類似性が高いためであると考えられる。

全てを畳み込み結合により実現している U-Net では，切り出した小さなサイズの学習画像を学習することより，より広範囲の画像領域を分割することが出来た。これは，工業分野における画像処理のみならず，水産業，林業，農業，建築，防災など様々な分野への応用が期待される。但し，本実験の U-Net の汎化能力は低く，正確な領域分割のためには，できるだけ多くの種類の画像を学習する必要がある。しかし，過学習に陥らないように一般的に異なる画像の教師画像を十分に用意できるとは限らないため，入手可能な画像に対して，ランダム消去，明彩変更などの画像処理を適用した教師画像による増量が有効な手段であると考えられる。

文　　献

1) 福島 邦彦：神経回路と情報処理，朝倉書店(1989).
2) 廣川 勝久：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，深層学習による異常検知手法の簡単な比較（第 1 報）**35**（2022）.
3) 廣川 勝久：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，深層学習による異常検知手法の簡単な比較（第 2 報）**36**（2023）.
4) Olaf Ronneberger, Philipp Fischer, and Thomas Brox,: U-net: Convolutional networks for biomedical image segmentation, arXiv: **1505.04597**, (2015).
5) WWW：Visual Object Classes Challenge 2012 (VOC2012)，http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html
6) Speth, S., Gonçalves, A., Rigault, B., Suzuki, S., Bouazizi, M., Matsuo, Y. & Prendinger, H.: Deep Learning with RGB and Thermal Images onboard a Drone for Monitoring Operations. *Journal of Field Robotics*, **1-29**, (2022).