

指導教員印

## 2021年度 卒業論文

特徴量の抽象度に応じて受容野の範囲に  
制限を設けた Deformable convolution

指導教員 荒井 秀一 教授

東京都市大学 知識工学部  
情報科学科  
知識情報処理研究室  
1812035 折田 汐凪

# 目 次

<b>第1章 序論</b>	<b>1</b>
<b>第2章 背景</b>	<b>2</b>
2.1 Semantic segmentation . . . . .	2
2.2 従来研究 . . . . .	3
2.2.1 CNN(Convolutional Neural Network) . . . . .	3
2.2.2 Bottleneck 構造の CNN . . . . .	6
2.2.3 HRNet (High-Resolution Network) . . . . .	6
2.2.4 DHRNet (Deformable convolution High-Resolution Network) . . . . .	7
2.3 研究目的 . . . . .	8
2.3.1 受容野の範囲 . . . . .	9
2.3.2 中心画素の offset . . . . .	9
<b>第3章 提案手法</b>	<b>10</b>
3.1 DHRNet (Deformable convolution High-Resolution Network) . . . . .	10
3.1.1 Basic Block . . . . .	11
3.1.2 Bottleneck . . . . .	12
3.1.3 Exchange Unit . . . . .	13
3.1.4 Up sampling . . . . .	13
3.1.5 Deformable Basic Block . . . . .	14
3.2 Deformable Convolution . . . . .	14
3.2.1 offset . . . . .	15
3.3 受容野の範囲に制限を設ける活性化関数の提案 . . . . .	16
3.3.1 DHRNet における特徴量の広がり . . . . .	17
3.3.2 Hard tanh7 . . . . .	18
3.4 offset の中心画素を固定する提案 . . . . .	19
<b>第4章 実験</b>	<b>21</b>
4.1 実験条件 . . . . .	21
4.1.1 データセット . . . . .	21
4.1.2 評価指標 . . . . .	21
4.1.3 ハイパーパラメタ . . . . .	22
4.1.4 実験環境 . . . . .	22

4.2 実験結果及び検討 . . . . .	23
4.2.1 DHRNet の有効性の確認 . . . . .	23
4.2.2 提案手法の有効性の確認 . . . . .	25
<b>第5章 結論</b>	<b>28</b>
<b>参考文献</b>	<b>30</b>
<b>付録A 基礎事項</b>	<b>33</b>
A.1 活性化関数 . . . . .	33
A.2 Batch Normalization . . . . .	34
A.3 バイリニア補間 . . . . .	34
A.4 $1 \times 1$ 置き込み (Pointwise Convolution) . . . . .	35
A.5 誤差逆伝播法 (Back-propagation) . . . . .	35
A.6 損失関数 . . . . .	35

# 第1章

## 序論

近年の少子高齢化による労働人口の減少により、製造業界や物流業界では十分に労働者を確保することが困難になっている。人手不足を解決するために、ロボットなどを用いることで、人が行っている作業を自動化させることが取り組まれている。自動化の例として、倉庫での倉庫での荷積みや荷下ろし、ピッキング作業などが挙げられる。これらの作業はものを掴む、運ぶ、離す動作が必要であり、対象の物体の正確な位置と形状を把握できないと対象の物体や周辺の積荷などを破損してしまう可能性がある。今までに研究されていた“物体検出”[1][2]は画像中にある物体のクラスと位置を認識して、矩形の枠で囲む技術であった。そのため、大まかな物体の位置は把握できても物体の形状までは認識できていなかった。この問題を解決するために“Semantic Segmentation”[3][4]が登場した。Semantic Segmentationは入力画像に対してピクセル単位でクラス分類を行うことで、物体の輪郭で領域分割する技術である。これにより画像中の物体のクラス、位置のみならず、詳細な形状まで認識することができるため、Semantic Segmentationはロボットビジョンへの応用が期待されている。このような分野に需要があるため、画像認識分野においてSemantic Segmentationに関する研究が盛んに行われている。

# 第2章

## 背景

Semantic segmentataion では, 深層学習を用いた手法が多く提案されている. その中でも CNN(Convolutional Neural Network)[5] を用いた手法により認識精度は飛躍的に向上した. そのことから, 今までに CNN をもとにした手法は数多く提案され, アーキテクチャが改良されてきた. そして近年, convolution を改良した deformable convolution[6] を用いたことにより, さらに認識精度が向上した. しかし, deformable convolution の挙動は調査されていない.

本章では, まず 2.1 節で semantic segmentation を概説することにより本研究分野の立ち位置を明確にする. その上で, 2.2 節で semantic segmentation において現在までに提案してきた CNN をもとにした従来手法を紹介し, 従来手法のネットワークに共通する構造上の問題点を明らかにする. 次に, 2.2.3 項で従来手法の問題点を改善した手法を取り上げ, 最後に 2.3 節でこれらの従来手法の問題点を明らかにしたうえで, 本論文の研究目的を定義する.

### 2.1 Semantic segmentation

Semantic segmentation は入力画像に対してピクセル単位でクラス分類を行うことで, 画像中の様々な物体をそれぞれの外形で領域分割する技術である [3][4]. 図 2.1 中の上段は入力画像であり, 下段に示すクラスごとに入力画像を分離した結果が中段の画像である. そのため, semantic segmentation は画像中に存在する様々な物体が, それぞれどのクラスに属し, どのような形状で, どの位置に存在するのかを表現可能であり, シーン理解のタスクにおいて重要な手法である. 近年では, 深層学習を用いた手法の提案により, 以前よりさらに高精度な推論が可能になった [7].

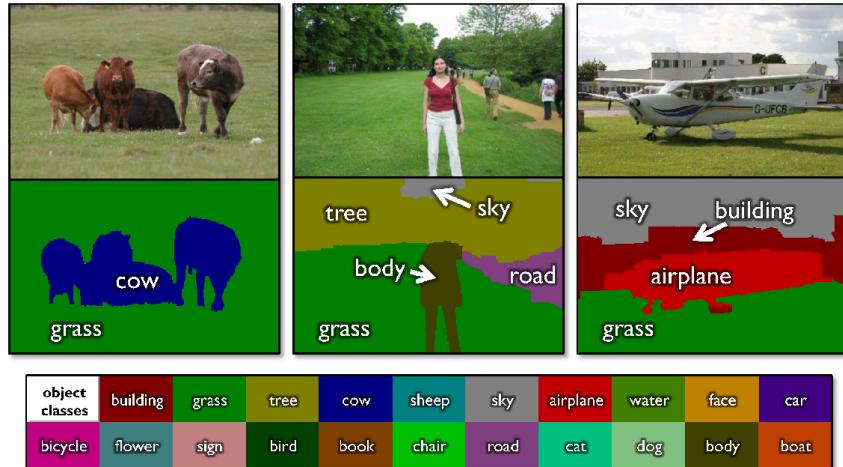


図 2.1: Semantic Segmentation の例 (上段:入力画像, 中段:分類結果, 下段:クラス凡例)[4]

## 2.2 従来研究

Semantic segmentation は深層学習を用いた手法が多く提案されてきた。それらの多くは CNN を元としている。本節では、多くの手法の元となっている CNN を説明した後に、CNN を用いた手法の代表例である Bottleneck 構造の CNN を紹介する。その後に従来の Bottleneck 構造の CNN より高精度なセグメンテーションを可能にした HRNet (High-Resolution Network)[8] を説明した上で、CNN に変更を加えることで HRNet を改良した DHRNet (Deformable High-Resolution Network)[9] を説明する。

### 2.2.1 CNN(Convolutional Neural Network)

画像認識では主に CNN[5] が用いられてきた。CNN は図 2.2 に示すような深層学習に用いられる多層ニューラルネットワークの 1 つである。図 2.2 では入力画像を畠み込み層 (convolution layer), プーリング層 (pooling layer) で特微量抽出した後、一般的な全結合ネットワークがもつ全結合層 (full connection layer) に通することでクラス分類をおこなっている。ネットワークを多層にすることで、入力に近い層ではエッジなどの特徴を抽出し、深い層になるにつれ、より抽象的な特徴を抽出できるように構成されている。semantic segmentation では、ネットワークの出力が CNN の全結合層を畠み込み層に置き換えた、畠み込み層、プーリング層 のみで構成される FCN(Fully Convolutional Network)[7] をベースとしたアーキテクチャが多く提案されている。

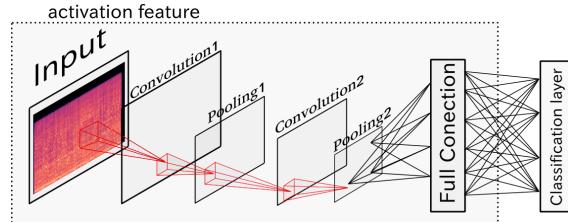


図 2.2: Convolutional Neural Network の例

### 畠み込み (Convolution)

畠み込みは、入力データと学習した重みを持つフィルタとで畠み込み演算をして、“特徴マップ”を出力する処理である。畠み込み演算は、図 2.3 の灰色で表したフィルタを一定間隔でスライドさせながら、フィルタの要素と入力データの対応する要素を乗算し、その和を求める演算である。また畠み込みはフィルタの数、フィルタのサイズ、ストライド、パディングという設定する必要のあるパラメタを持つ。

まず、フィルタの数があり、設定したフィルタの数だけ特徴マップを出力する。

次にフィルタのサイズはフィルタの幅と高さであり、図 2.3 のカーネルは高さ 3、幅 3 である。

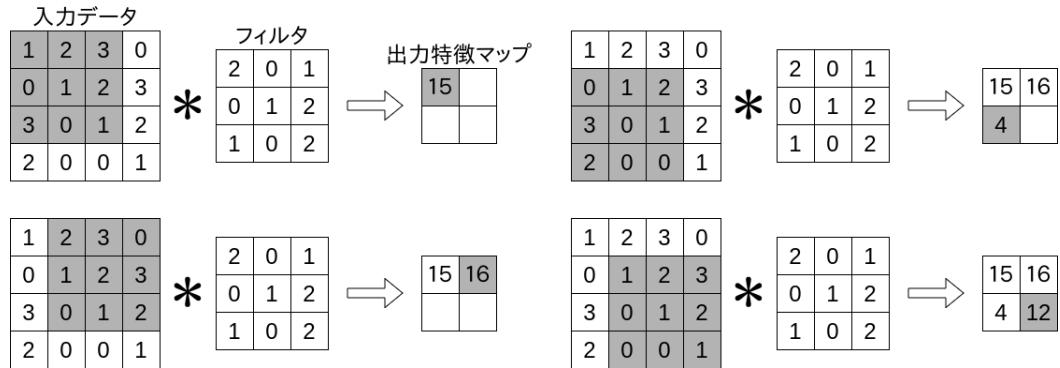


図 2.3: Convolution の例 (\*は畠み込み積分)

入力データを  $I(x, y)$ 、出力特徴マップを  $O(x, y)$ 、フィルタカーネルを  $h(m, n)$ 、フィルタサイズを  $(2M + 1) \times (2N + 1)$  とした場合、畠み込みは式 (2.1) のように書ける。

$$O(x, y) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} h(m, n) I(x + m, y + n) \quad (2.1)$$

入力データ  $I(x, y)$  のサイズを  $(X, Y)$  とすると、出力特徴マップ  $O(x, y)$  は  $(M \leq x < X - M, N \leq y < Y - N)$  の範囲までしか求められず、出力特徴マップのサイズが小さくなってしまう。これを防ぐためにはパディングという処理を行う。パディングは、図 2.4 のように入力データの周囲を一定の値で埋める処理である。パディングにより出力特徴マップ

$O(x, y)$  は  $(0 \leq x < X, 0 \leq y < Y)$  のように入力データと同じサイズの出力特徴マップになる。

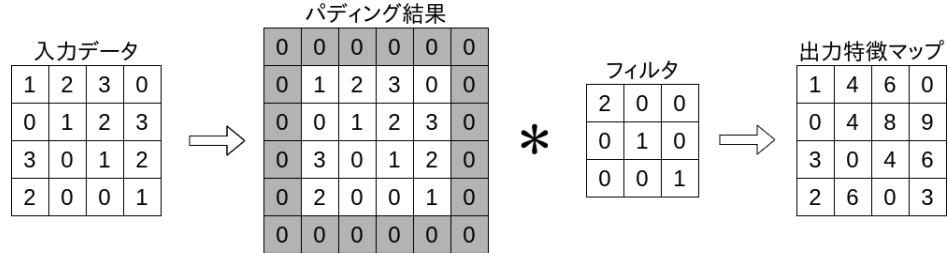


図 2.4: パディングの例

ストライドはフィルタカーネルを移動させる間隔を決めるパラメータである。パディングが入力データと出力特徴マップのサイズを合わせるための処理であるのに対し、ストライドは 1 より大きい整数に変更することで出力特徴マップのサイズを縮小させることができる。

ストライドを 1 より大きい整数  $s$  に変更した Convolution は式 (2.2) のように書ける。

$$O(x, y) = \sum_{m=-M}^M \sum_{n=-N}^N h(m, n) I(sx + m, sy + n) \quad (2.2)$$

これにより  $O(x, y)$  は  $(0 \leq x < \frac{X}{s}, 0 \leq y < \frac{Y}{s})$  のように出力特徴マップのサイズが小さくなる。しかし  $s$  を大きくしていくと、それに伴い特徴を見落とす可能性が高くなる。特に  $s$  がフィルタカーネルの高さ、または幅を超えてしまうと明らかにフィルタに覆われない領域が出現し、特徴を見落としてしまう。そのため、特徴を見落とさないためには、ストライド  $s$  をフィルタカーネルの高さ、幅より小さいサイズに収めるか、後述するプーリングにより出力特徴マップを小さくする方法がある。

## プーリング (Pooling)

プーリングは、畳み込みにより得られた特徴マップのサイズを縮小する処理である。この処理はパラメータの削減や、受容野を広げる役割があり、また微小な位置ずれに対しロバストになる。領域内の最大値を得る Max Pooling(図 2.5) や領域内の平均を得る Average Pooling などがある。ストライド 2 の畳み込みとストライド 1 の畳み込み +  $2 \times 2$  Pooling は出力特徴マップのサイズという面では同じであるが、前者が畳み込みの間隔を大きくして特徴マップを小さくしただけなのに対し、後者は特徴量抽出を行なった後により値が大きい特徴を残して抽象化するという面で、後者の方が処理の意味付けが明確にできる。

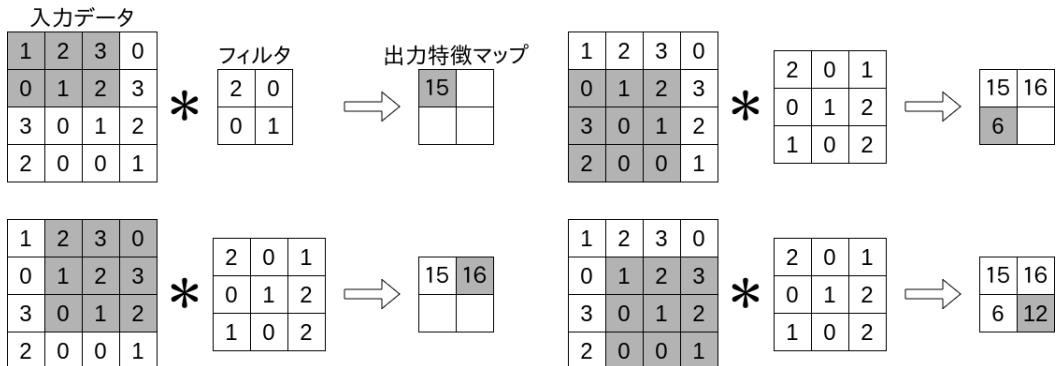


図 2.5: Max Pooling の例

## 2.2.2 Bottleneck 構造の CNN

Semantic segmentation の手法として, bottleneck 構造の CNN が多く用いられている。bottleneck 構造の CNN は, 入力画像を特徴量抽出しつつ, 解像度を下げていき, 学習可能な層などを用いて入力画像と同等の解像度まで復元する構造をもっている。bottleneck 構造の CNN の例を図 2.6 に示す。図 2.6 のエンコーダ部分では特徴量を抽象化していく, 低解像度になった特徴マップを図 2.6 のデコーダー部分で特徴抽出をしつつ, 高解像度化している。

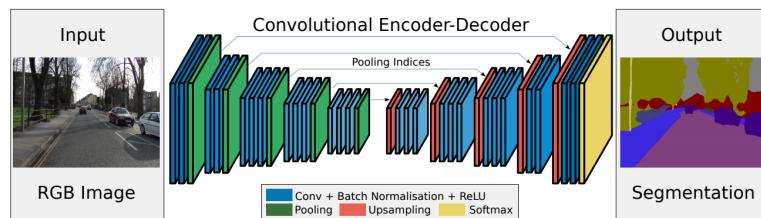


図 2.6: bottleneck 構造の CNN(SegNet[10])

bottleneck 構造の CNN の代表的なアーキテクチャには, “SegNet”[10], “U-Net”[11], “RefineNet”[12] などがある。これらの手法では, 入力に近い部分で抽出した特徴マップを一旦抽象化してしまうため, 高解像度の特徴マップがもつ物体の詳細な形状の情報が失われてしまう。これは物体の形状を正確に認識する必要がある semantic segmentation において大きな問題である。

## 2.2.3 HRNet (High-Resolution Network)

HRNet[8] は, 高解像度と低解像度の特徴マップを直列に扱う従来の手法とは異なり, ネットワーク全体を通じ高解像度の特徴マップを維持できる構造を持っている。高解像度の特徴マップのみでは抽象的な特徴は抽出できないため, 層が深くなるごとに低解像度化した

ネットワークを追加することで、図 2.7 に示す最上段の灰色にあたる高解像度部分から最下段の赤色にあたる低解像度までの特徴を並列に処理できる構造になっている。この構造により HRNet は、bottleneck 構造の CNN の構造上の問題である、物体の細かな形状の情報を持てない点を改善した。

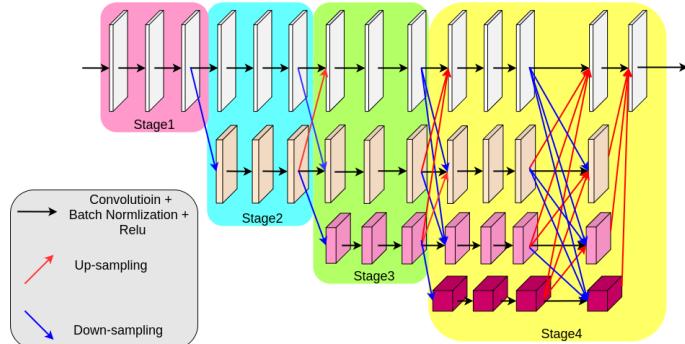


図 2.7: HRNet

#### 2.2.4 DHRNet (Deformable convolution High-Resolution Network)

DHRNet[9] は HRNet の低解像度部分にある畳み込み層を受容野が変形可能な deformable convolution に置き換えたものである。前項までに semantic segmentation の CNN を用いた代表的な手法を紹介してきたが、根本となる畳み込み層を改善する試みはされていなかった。畳み込みは 2.2.1 項で述べたように、入力画像に重みを持った固定サイズのフィルターカーネルをかけ特徴を抽出する処理である。そのため、入力特徴マップのうちある出力特徴量に影響を及ぼす範囲である受容野は固定である。しかし、semantic segmentation では、様々な形状や大きさの物体の特徴を抽出する必要があるため、固定形状の受容野では特徴抽出に適切ではない。この問題を解決するために、DHRNet は deformable convolution という畳み込みを HRNet に適応した[9]。deformable convolution は、追加の畳み込み層で学習した offset ベクトルを用いることで、受容野を可変形状にする畳み込み層である。図 2.8 に受容野を可変にしたフィルターカーネルの例を示す。図 2.8 の赤点が元の畳み込みの受容野、赤線は offset ベクトル、そして青点が offset により移動した後の受容野を表している。受容野が offset ベクトルによって可変になっていることがわかる。DHRNet は都市シーンのセグメンテーションのタスクなどで優秀な成績を収めている。この手法に関する詳しい説明は 3 章で述べる。

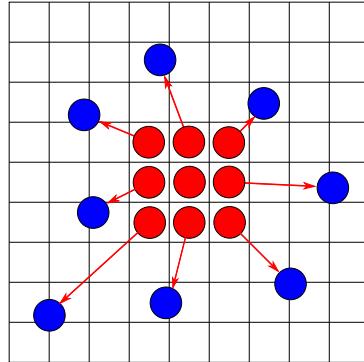
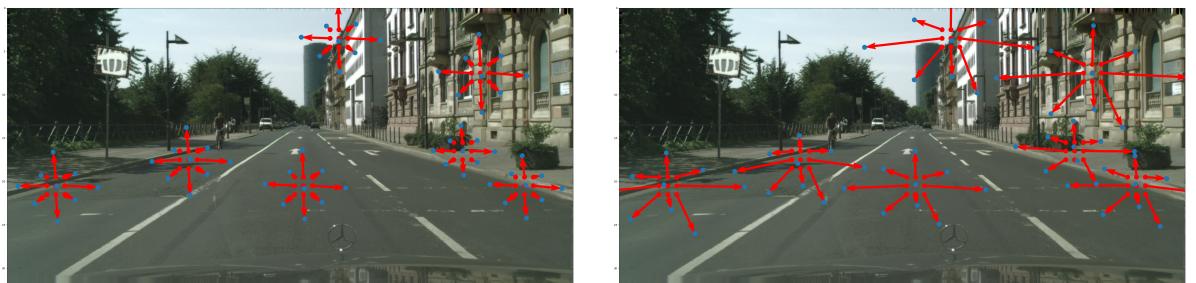


図 2.8: offset による受容野の変化例

## 2.3 研究目的

2.2.4 節で紹介した DHRNet は, HRNet と比較して識別精度を向上させるさせることに成功した. しかし, なぜ deformable convolution を導入したことでの認識精度が向上したのかは明らかにされていない. そこで, deformable convolution の挙動を明らかにするために, deformable convolution により変化したフィルタカーネルの位置を可視化し受容野の変化を視覚的に確認した. DHRNet には deformable convolution 層が 2 層含まれており, 図 2.3(a) は 1 層目, 図 2.3(b) は 2 層目の可視化結果である. 可視化にあたって offset の変化を評価するために 7 つの画素を選んだ. 前提として, offset の適切な変化量を定義することは困難であるため, 受容野が必ず異なる形状であるべきだと考えた評価点を選定している. 仮説としては, エッジ付近と物体の中心付近の受容野の形状には明確な違いが現れると考えたため, 評価の対象とする画素は, 道路と歩道とのエッジ付近や, 物体の中心付近とした.

可視化結果である図 2.3(a) と 図 2.3(b) を見ると受容野の大きさが変化していたが, 評価



(a) 1 層目の deformable convolution の offset    (b) 2 層目の deformable convolution の offset  
図 2.9: deformable convolution による受容野の変化を可視化

画素ごとに受容野の形状を比較すると, 変化が乏しいことが確認できた. さらに, 図 2.3 の各サンプリング点の左右方向の offset の大きさに注目すると, 明らかに offset ベクトルが大きいものが存在する. この可視化結果より deformable convolution の問題点を 2 つ考察した.

### 2.3.1 受容野の範囲

畳み込みは重みを持ったフィルタカーネルと、それと同じサイズの入力特徴量とを掛け合わせる処理であるため、出力特徴量が表現できるのは畳み込まれた局所的な特徴量の範囲内に制限される。つまり畳み込みは限られた範囲の中で特徴抽出を繰り返す処理である。ここで、deformable convolution の offset は畳み込み層の出力特徴量であるため、畳み込み層の入力特徴量が持つ情報の範囲内のみ表現可能である。そのため、deformable convolution の受容野の範囲は入力特徴量がもつ情報の範囲に制限されるべきである。しかし、理由は後述するが、現状の deformable convolution のアーキテクチャは受容野の範囲を制限する機構を持っていない。よって、本研究では DHRNet の deformable convolution における適切な需要野の範囲を検討し、受容野の広がりを制限する機構を追加することで受容野を適切な範囲に収めることを 1 つ目の研究目的とする。

### 2.3.2 中心画素の offset

## 第3章

### 提案手法

本研究では 2.3 節で述べたように, DHRNet の deformable convolution 部分に受容野の広がりを制限する機構を追加することで, 受容野を適切な範囲に収めることを目指す. そのために, DHRNet の構造を詳しく説明した後に, offset の範囲に制限を設ける新たな活性化関数を提案する.

#### 3.1 DHRNet (Deformable convolution High-Resolution Network)

DHRNet はネットワーク全体を通して高解像度を維持するアーキテクチャである. 図 3.1 に示すように HRNet は異なる解像度で処理をするネットワークを組み合わせた 4 つのステージで構成されている. ステージ 1 は入力と同等の解像度で処理するネットワークのみで構成される. 続くステージ 2 ではステージ 1 のネットワークに, 入力の  $1/2$  倍スケールの低解像度を処理するネットワークを追加する. さらに続くステージ 3 では  $1/4$  倍スケール低解像度を処理するネットワークを追加し, 最後のステージ 4 では  $1/8$  倍スケール低解像度を処理するネットワークを追加する. また, ステージ 4 には deformable convolution が適応されている. そして解像度を下げるたびにチャンネル数は 2 倍になる (16, 32, 64, 128 チャンネル). 以降, 図 3.1 に示した本アーキテクチャの詳細について説明していく. また, 図中の ReLU, Batch Normalization については, それぞれ, 付録 A.1, 付録 A.2 で説明する.

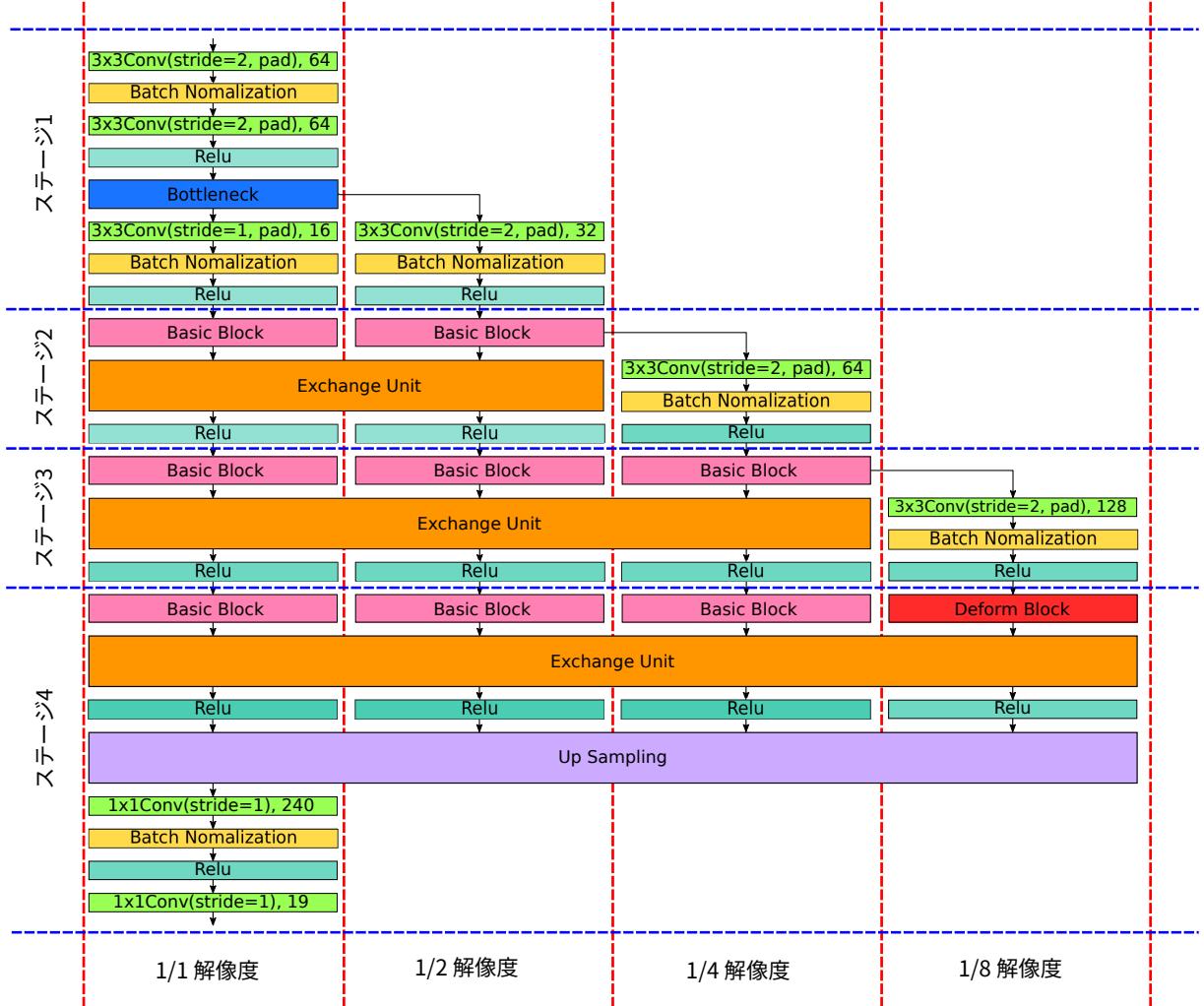
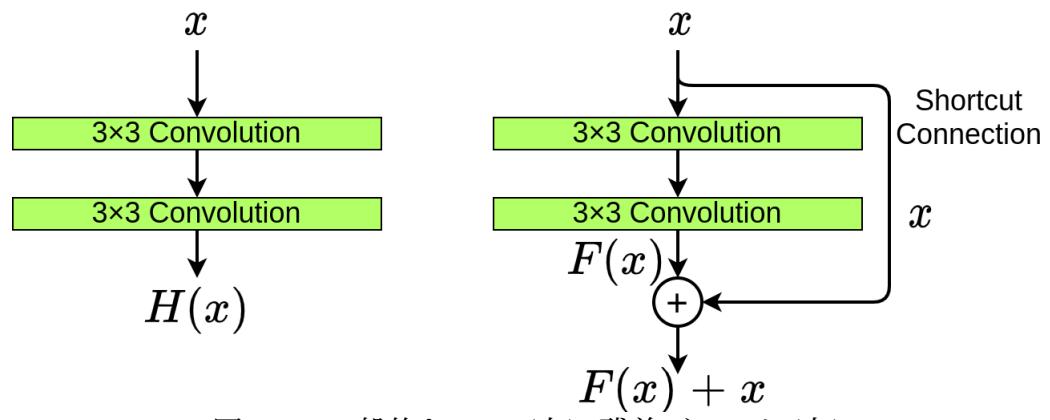
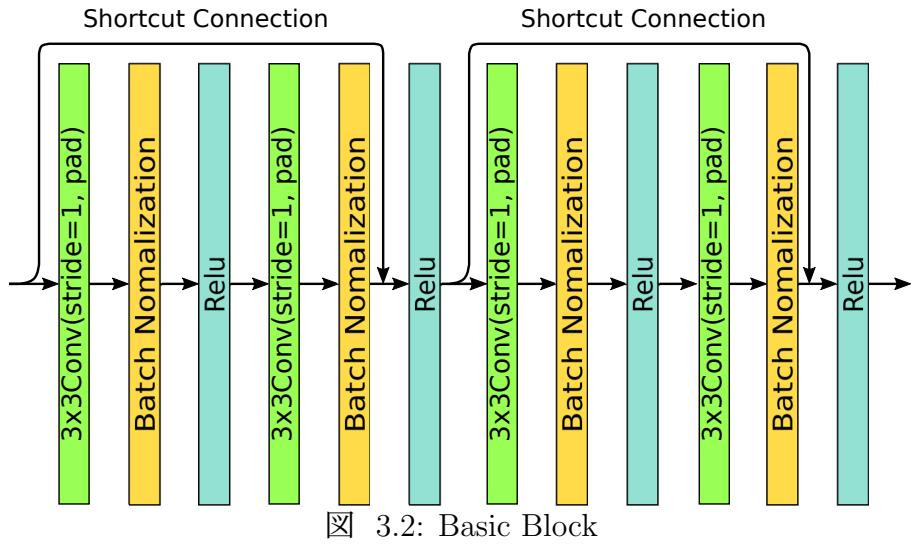


図 3.1: DHRNet の構造

### 3.1.1 Basic Block

図 3.1 のピンク色で示した “Basic Block” の構造を示したものが図 3.2 である。このブロックは HRNet において特徴抽出を担う重要な箇所である。一般にこのような構成で層を深くすると、いわゆる “勾配消失” により、ネットワークの重みが更新できなくなる。これを解決する方法はさまざま提案されているが、HRNet では “ResNet(Residual Network)” [13] で提案された “残差ブロック” を用いている。図 3.3 に示すように、一般的な CNN は入力  $x$  を数段の Convolution 層に通して  $H(x)$  を得るのに対し、残差ブロックでは、CNN の出力  $F(x)$  と入力  $s$  との和を出力  $H(x) = F(x) + s$  とする。これにより、CNN では  $F(x) = H(x) - x$  なる  $H(x)$  と入力  $s$  との残差を学習すればよく、残差信号の平均値は 0 に近いので勾配消失を防ぐ効果が期待できる。



### 3.1.2 Bottleneck

図 3.1 の青色で示した “Bottleneck” は Basic Block と同様, HRNetにおいて特徴抽出を担う箇所で、図 3.4(右)に示すような構造である。構造は残差ブロック(図 3.4(左))に似ているが、Bottleneck では入力のチャンネル数を  $1 \times 1$ 畳み込み(付録 A.4)を用いて小さくしてから  $3 \times 3$ 畳み込みで特徴抽出し、最後の  $1 \times 1$ 畳み込みでチャンネル数を復元する。これにより計算コストを削減できる。例えば、図 3.4において残差ブロックの入力チャンネル 64、Bottleneck が入力チャンネル数 256 と Bottleneck の方がチャンネル数が大きいが、これらの計算コストは同等である。

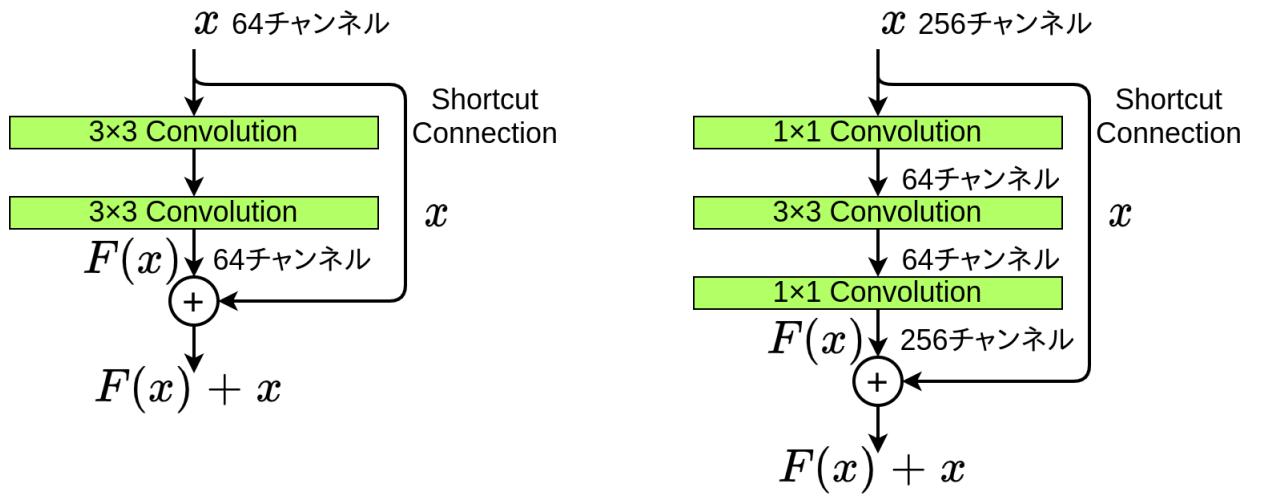


図 3.4: 残差ブロック(右), Bottleneck(左)

### 3.1.3 Exchange Unit

HRNet では異なる解像度の特徴マップ間で情報共有するために “Exchange Unit” がある。図 3.1 ではオレンジ色の部分がそれにあたる。具体的に, Exchange Unit では異なる解像度の特徴マップ間でバイリニア補間によるアップサンプリング、またはストライド 2 の  $3 \times 3$  畳み込みによるダウンサンプリングで特徴マップのサイズを合わせ、特徴マップ同士を加算する。これにより異なる解像度の特徴マップ間での情報共有を可能にしている。

### 3.1.4 Up sampling

図 3.1 の紫色で示した Up sampling は  $1/2$ ,  $1/4$ ,  $1/8$  の低解像度の特徴マップをバイリニア補間でアップサンプリングし、それらをチャンネル方向を軸にして  $1/1$  解像度の特徴マップに連結する。この Up sampling ブロックで得られた特徴マップを分類に用いる。

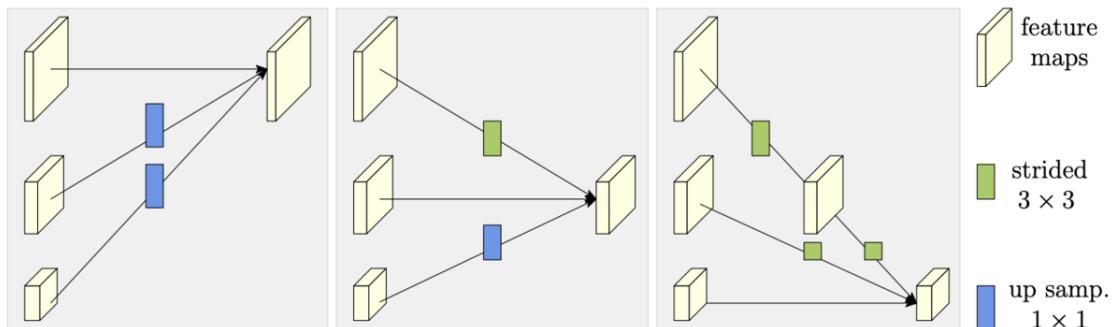


図 3.5: exchange unit[8]

### 3.1.5 Deformable Basic Block

図 3.1 の赤色で示した “Deformable Basic Block” の構造を示したものが図 3.6 である。 “Deformable Basic Block” は、Basic Block の入力から数えて 1 番目と 3 番目の  $3 \times 3$ Convolution を deformable convolution に変更した

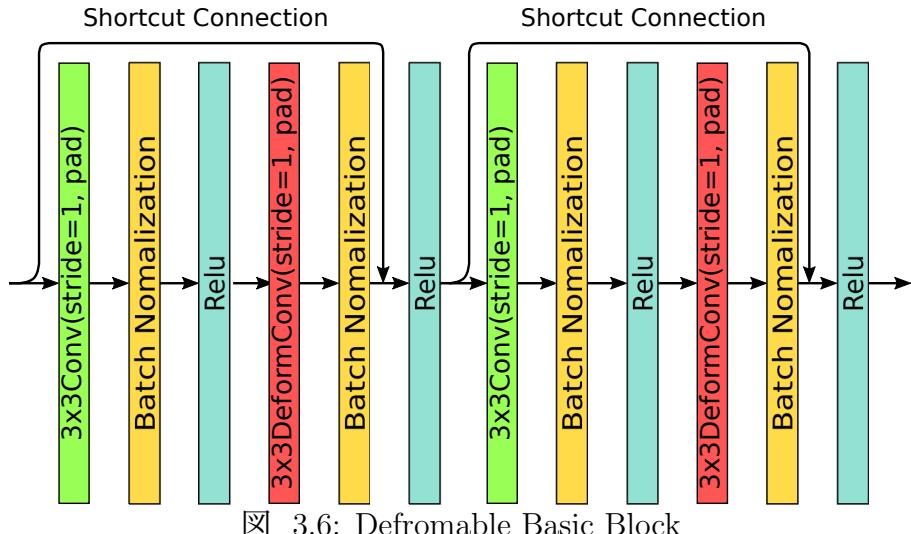


図 3.6: Defromable Basic Block

## 3.2 Deformable Convolution

deformable convolution 署み込みのサンプリング位置からの変位である “offset” を計算、学習をしてフィルタのサンプリング位置を動的に変化させる。そのため画像中の物体のスケール、形状に合わせて受容野のサイズ、形状を変えることが可能である。

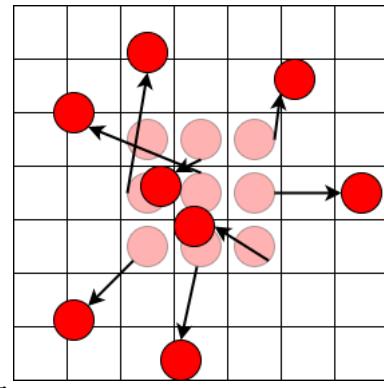


図 3.7: Deformable Convolution

署み込みのサンプリング位置  $p_n$  の集合を  $\mathcal{R}$ ,  $w$  を重み,  $x$  を入力特徴マップ,  $y$  を出力特徴マップと表したとき, 出力特徴量マップ  $y$  の各位置  $p_0$  に対応する Convolution は式(

3.1) のように書ける.

$$R = \{(-1, -1), (-1, 0), \dots, (0, 1), (1, 1)\}$$

$$y(p_0) = \sum_{p_n \in R} w(p_n) x(p_0 + p_n) \quad (3.1)$$

deformable convolution では、式 (3.2) に示すように convolution のサンプリング位置  $p_0 + p_n$  に offset  $\Delta p$  を加算することで受容野を可変にしている。

$$y(p_0) = \sum_{p_n \in R} w(p_n) x(p_0 + p_n + \Delta p_n) \quad (3.2)$$

### 3.2.1 offset

offset の算出方法を図 3.8 に示す。offset の算出では、まず入力特徴マップを図 3.8 中の緑の枠線で示す追加の convolution に通すことで、緑の立方体で示す入力特徴量の全てのサンプリング点の offset である offset field を出力する。offset は  $x$  方向、 $y$  方向の移動量を示す 2 次元ベクトルであるため、offset field のチャンネル数は  $2N$  チャンネルである ( $N$  はフィルタカーネルのサイズで、 $3 \times 3$  フィルタカーネルの場合  $N = 9$ )。次に offset field から 各サンプリング点の offset を取り出すことで、図 3.8 中の緑の格子で示す offset を決定する。そして、図 3.8 中の青い点線で示すように、offset で convolution の受容野を変形する。しかし、offset は整数であるとは限らないため、offset によって変化した後のサンプリング位置が非整数座標になってしまう場合がある。その場合、非整数座標に画素値は存在しないため、バイリニア補間で非整数座標の画素値を算出する必要がある(式 (3.3))。( $G$  はバイリニア補間、 $q$  は  $p_0 + p_n + \Delta p_n$  の周辺の整数座標)

$$x(p_0 + p_n + \Delta p_n) = \sum_q G(q, p_0 + p_n + \Delta p_n) x(q) \quad (3.3)$$

offset  $\Delta p_n$  を学習するために、式 (3.2)、(3.3) のバイリニア補間をもとに勾配を求める式 (3.4) のように求まる。これを用いて、誤差逆伝播法(付録 A.5)により学習を行う。

$$\begin{aligned} \frac{\partial y(p_0)}{\partial \Delta p_n} &= \sum_{p_n \in R} w(p_n) \frac{\partial x(p_0 + p_n + \Delta p_n)}{\partial \Delta p_n} \\ &= \sum_{p_n \in R} [w(p_n) \sum_q \frac{\partial G(q, p_0 + p_n + \Delta p_n)}{\partial \Delta p_n} x(q)] \end{aligned} \quad (3.4)$$

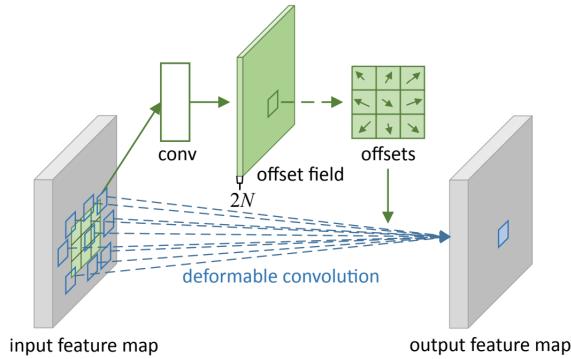
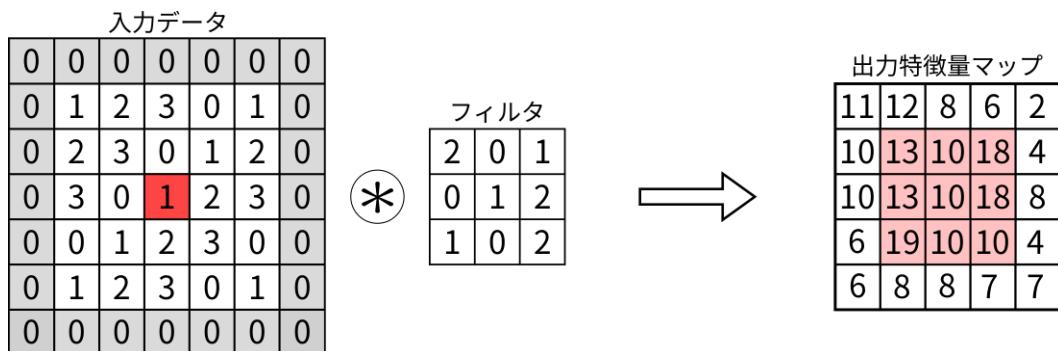


図 3.8: Deformable Convolution の流れ [6]

### 3.3 受容野の範囲に制限を設ける活性化関数の提案

deformable convolution の offset によって決定する受容野が収まるべき範囲は、畳み込み層を重ねたときの、ある入力画像の画素が影響を及ぼす範囲によって決定できると考えた。例えばストライド 1, padding1 の  $3 \times 3$  畳み込みの場合は、1 つの画素が最大で  $3 \times 3$  の範囲の出力特徴量に影響を及ぼす。図 3.9 に、ストライド 1, padding1 の  $3 \times 3$  畳み込みにより入力特徴量の影響を及ぼす範囲が広がる例を示す。図 3.9 中にある入力データの灰色は padding であり、白は padding 前の元の入力データである。そして、入力データ内の赤色で示す特徴量を入力として、 $3 \times 3$  畳み込みにより得られた出力特徴量が出力特徴量マップの薄い赤色部分である。さらに、downsampling の際にも各特徴量が影響を及ぼす範囲は広がる。この入力特徴量が影響を及ぼす範囲内に受容野が収まれば、offset により変化したサンプリング点に変化前のサンプリング点の特徴が必ず含まれるようになる。

図 3.9: ストライド 1, padding1 の  $3 \times 3$  畳み込み層により特徴量の影響を及ぼす範囲が広がる例

しかし、現状の deformable convolution の offset は図 3.9 中の緑色の枠に示す畳み込み層の出力であり、値の範囲を制限をせずに受容野の変更に用いている。そのため、deformable convolution は入力特徴量が影響を及ぼす範囲外まで受容野を拡大することが可能な構造を

している。そこで、受容野の範囲を特徴量の影響を及ぼす範囲に制限するために、DHRNet の低解像度部分における特徴量が影響を及ぼす範囲を確認した。

### 3.3.1 DHRNet における特徴量の広がり

図??より、DHRNet の入力画像はまずストライド 2, padding1 の  $3 \times 3$  畳み込み層を 2 層通る。このストライド 2, padding1 の  $3 \times 3$  畳み込み層では、出力特徴マップのサイズが入力特徴マップの  $1/2$  になるのに加え、ある入力特徴量が影響を及ぼす範囲が出力特徴マップの  $2 \times 2$  の範囲まで広がる。図 3.10 は、 $8 \times 8$  の入力特徴マップをストライド 2, padding1 の  $3 \times 3$  畳み込み層で特徴抽出したときに特徴量の影響を及ぼす範囲が広がる例である。図 3.10 の入力画像の赤色で示す入力画素は、出力特徴量マップの薄い赤色で示す  $2 \times 2$  の範囲の算出に用いられる。そして、2 層目の畳み込み層では特徴マップのサイズは  $1/2$  になるが、ある画素が影響を及ぼす範囲は  $2 \times 2$  のままである。

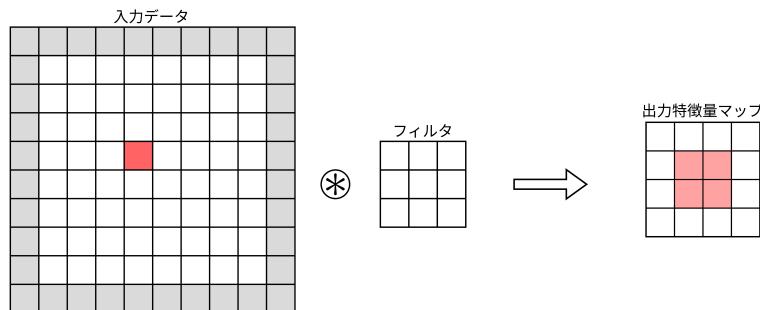


図 3.10: ストライド 2, padding1 の  $3 \times 3$  畳み込み層により特徴量の影響を及ぼす範囲が広がる例

その後、bottleneck で  $1 \times 1$ ,  $3 \times 3$ ,  $1 \times 1$  の畳み込み層を通る。 $1 \times 1$  畳み込み層はチャンネル方向のみ畳み込むため、特徴量の影響を及ぼす範囲は変化しない。bottleneck の  $3 \times 3$  畳み込みはストライド 1, padding1 であるため、図 3.11 の赤色で示す  $2 \times 2$  の範囲の特徴量が出力特徴マップの薄い赤色で示す  $4 \times 4$  の範囲まで広がる。

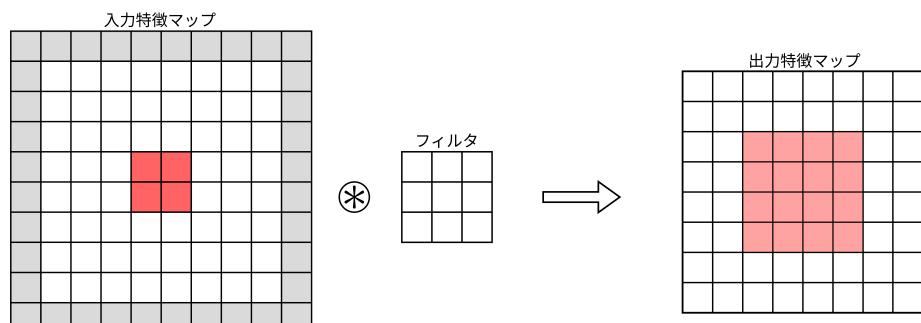


図 3.11: 入力特徴マップの  $2 \times 2$  の特徴量が影響を及ぼす範囲が、ストライド 1, padding1 の  $3 \times 3$  畳み込み層により広がる例

このように DHRNet でのある入力画素の影響を及ぼす範囲を計算したところ、低解像度部分の特徴マップでは  $13 \times 13$  の範囲まで広がっていた。ここで、deformable convolution の offset を算出する畠込み層のカーネルサイズは  $3 \times 3$  であるため、offset には  $15 \times 15$  の範囲の特徴量が影響を及ぼす。つまり、offset の範囲は上下左右 7 画素の範囲に制限すべきである。

### 3.3.2 Hard tanh7

本研究では、算出された offset の範囲を制限する hard tanh7 を offset に適応することを提案する。提案する hard tanh7 は、hard tanh[14] を拡張した活性化関数である。hard tanh 関数は図 3.12 に示すように、入力値が -1 より小さい場合と 1 より大きい場合は出力値が常に 0 であり、-1 以上、1 以内の範囲では入力値をそのまま出力する活性化関数であるが deformable convolution で制限したい値域とは異なる。

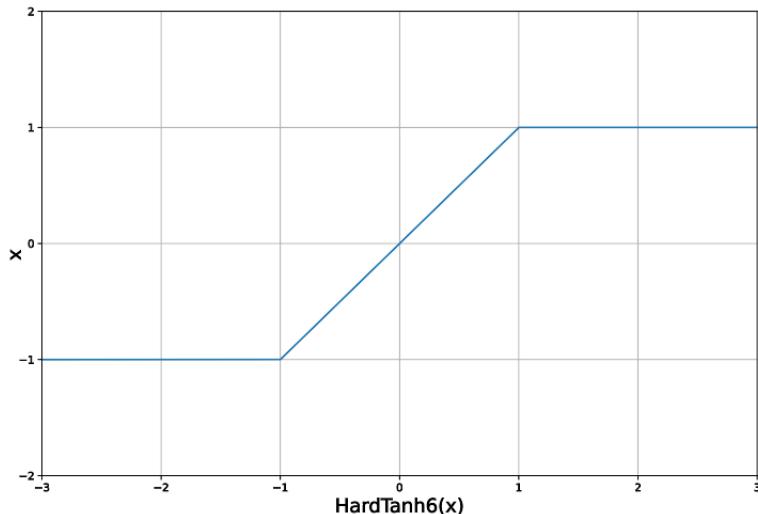


図 3.12: hard tanh 関数

そのため、提案する hard tanh7 は図 3.13 に示すように、hard tanh の値域を -7 以上、7 以下に拡張した。

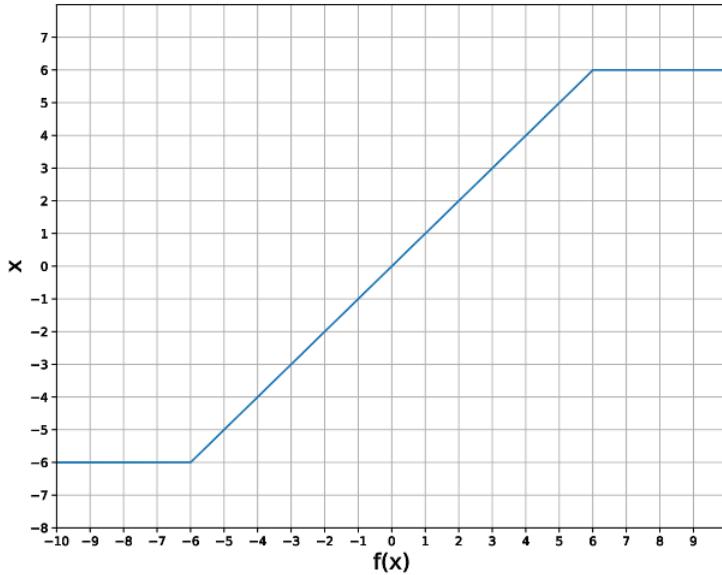


図 3.13: hard tanh7 関数

ここで, harh tanh7の入力を  $x$  としたときの式を式(3.5)に示す. 入力  $x$  が  $x < -7, x > 7$  ならば出力値  $\text{hardtanh7}(x) = 0$  であり,  $-7 \leq x \leq 7$  の範囲内ならば  $\text{hardtanh7}(x) = x$  である.

$$\text{hardtanh7}(x) = \begin{cases} 7(x > 7) \\ x(-7 \leq x \leq 7) \\ -7(x < -7) \end{cases} \quad (3.5)$$

この提案する活性化関数に offset を通することで, offset を特徴量が影響を及ぼす範囲内で学習させる.

### 3.4 offset の中心画素を固定する提案

現状のアーキテクチャでは畳み込みの中心サンプリング点も移動しており, 入力特徴量の中でサンプリングされない特徴が出てきてしまう可能性がある. 図 3.14 は padding1,  $5 \times 5$  サイズの入力特徴マップにおける deformable convolution のサンプリング位置の例を示す. 図 3.14 の左に示す入力特徴量を, 中央の緑色で表す可変形畳み込みカーネルで畳み込む場合, 右に示すように特徴マップでサンプリングに用いない箇所が存在する. それに加え, 畳み込みの位置普遍性が失われてしまう問題もあるため, 入力特徴量において特徴抽出に用いられない点があることは問題であると考えた.

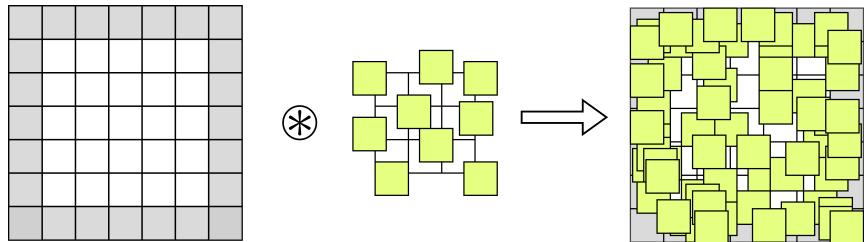


図 3.14: deformable convolution におけるサンプリング位置

そこで、本研究では全ての入力特徴量を特徴抽出に使用するために、deformable convolution の中心画素の offset を 0 に固定することを提案する。図??の中央に、deformable convolution の中心画素の offset を 0 に固定した可変形状の畠み込みカーネルを示す。畠み込みカーネルの中心サンプリング位置は強調するために赤色で表す。中心画素の offset を 0 に固定することで、図??の右に示すように畠み込みカーネルの中心サンプリング点で特徴マップのすべて特徴量をサンプリングすることができる。

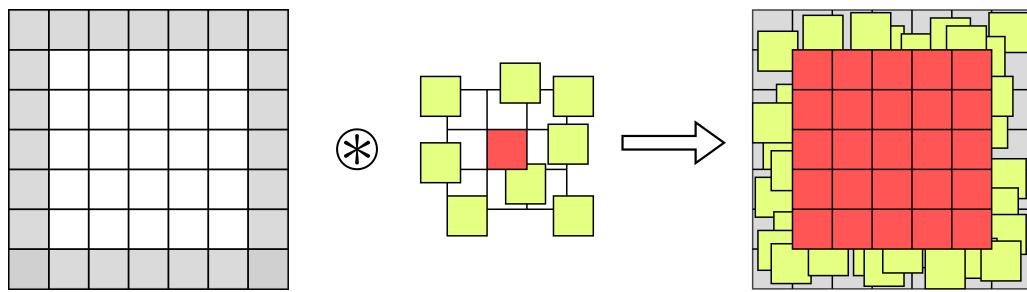


図 3.15: 中心画素の offset を 0 に固定した deformable convolution におけるサンプリング位置

# 第4章

## 実験

### 4.1 実験条件

本節では実験条件として、使用するデータセット、評価指標、ハイパーパラメータ、実験環境について述べる。

#### 4.1.1 データセット

本実験では“Cityscapes”[15]を使用した。Cityscapesは都市のストリートシーンの画像で構成されており、学習用画像が2975枚、検証用画像が500枚、テスト用画像が1525枚の計5000枚用意されている。学習の対象とするクラスはroad, sidewalk, building, wall, fence, pole, traffic light, traffic sign, vegetation, terrain, sky, person, rider, car, truck, bus, train, motorcycle, bicycleの19クラスである。

#### 4.1.2 評価指標

学習したモデルの性能を、スレットスコア(Threat Score)を用いた以下の2つの指標で評価する。

- IoU (Intersection over Union)
- MIoU (Mean Intersection over union)

スレットスコアは、正解データと予測結果を2値で比較し、その正誤を表4.1のような4つの状態で表すConfusion Matrixを用いて定義する。

表 4.1: Confusion Matrix

	正解はA	正解はAではない
Aと予測	TP(True Positive)	FP(False Positive)
Aではないと予測	FN(False Negative)	TN(True Negative)

True, False は正しく予測できたかを表し, Positive, Negative は正と負のどちらに予測されたかを表している. semantic segmentation において, True Positive は例えば A というクラスを正しく A と判定できたピクセル数, False Positive は別のクラスであるにもかかわらずクラス A と誤った判定したピクセル数, False Negative は A というクラスにもかかわらず別のクラスと誤った判定をしたピクセル数である.

次に, スレットスコアの計算式を示す. そもそもスレットスコアは気象分野で用いられる指標だが, ここで TP, FP をそれぞれ True Positive, False Positive のピクセル数とし, FN を False Negative のピクセル数として, semantic segmentation の評価値としてスレットスコアを用いたとき, このスコア値を IoU(Intersection over Union) と呼び, 式 (4.1) のように書ける. IoU はクラスごとに求まるが, これを全体のクラス数  $C$  で平均したものを MIoU(Mean Intersection over Union) と呼び, 式 (4.2) のように書ける.

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \quad (4.1)$$

$$\text{MIoU} = \frac{1}{C} \sum_{c=1}^C \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c} \quad (4.2)$$

### 4.1.3 ハイパーパラメタ

従来手法である HRNet[8] と比較するため, 同じハイパーパラメタを使用した. 使用したハイパーパラメタを表 4.2 に示す.

表 4.2: 使用したハイパーパラメタ

Batch Size	12
Epoch	484
活性化関数 (中間層)	ReLU
活性化関数 (出力層)	SoftMax

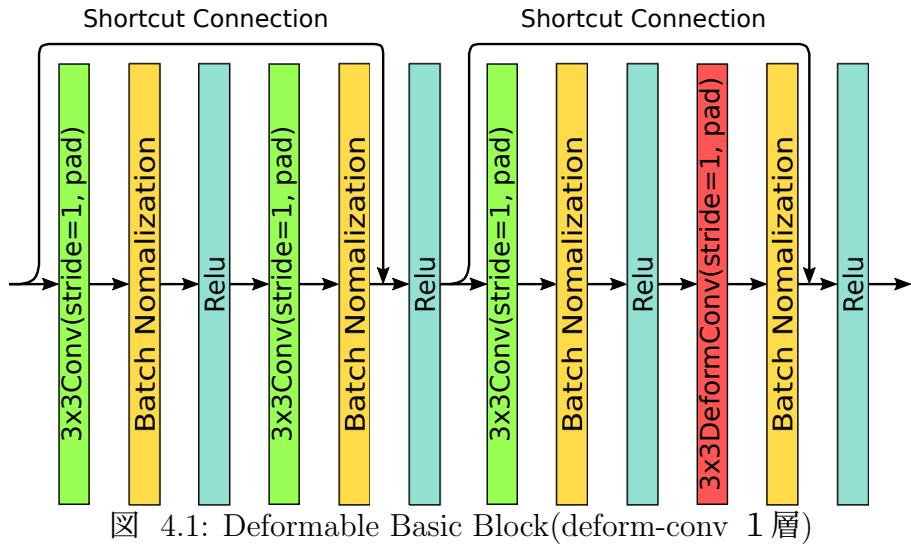
### 4.1.4 実験環境

本研究では以下の環境で実験を行った.

- Python 3.6.12
- Pytorch 1.5
- Tesla V100(32GB)

## 4.2 実験結果及び検討

本実験の目的の1つが deformable convolution の offset ベクトルの挙動を評価することである。しかし、deformable convolution が2つ含まれている既存の DHNet では、offset ベクトルの挙動を複合的に考える必要があるため、妥当な評価ができないと考えた。よって実験では、DHRNet の Deformable Basic Block を図 4.1 に示すような deformable convolution が1層のみに変更した DHRNet(deform-conv 1層)をベースにして実験を行う。



### 4.2.1 DHRNet の有効性の確認

本研究では、提案する活性化関数を offset に適応することの妥当性を検討する。そのためには、まず DHRNet(deform-conv 1 層)の有効性を確認する。これを実験 1 とする。HRNet と DHRNet(deform-conv 2 層), DHRNet(deform-conv 1 層)のクラス全体の結果を表 4.3 に示す。

表 4.3: 実験 1 の結果(クラス全体)

	MIoU(%)
HRNet	70.26
DHRNet(deform-conv 2 層)	<b>74.22</b>
DHRNet(deform-conv 1 層)	73.52

表 4.3 より、DHRnet(deform-conv) は DHRNet より 0.7pt 減少したが、HRNet からは 3.26pt 向上した。

次に、2.3 節で示した (1) 受容野の形状が物体によって変化しているように見えない。 (2) 受容野が影響を及ぼす範囲外まで広がっている。という問題点が DHRNet(deform 1 層)にも現れているかを確認するため、offset の可視化を図 4.2 に、offset の x 軸成分と y 軸成分

の分布を表したグラフを 4.3 と 4.4 に示す。4.3 と 4.4 の x 軸は offset の値であり、y 軸は頻度である。なお、赤い点線は offset ベクトルの存在するべき範囲を表す。

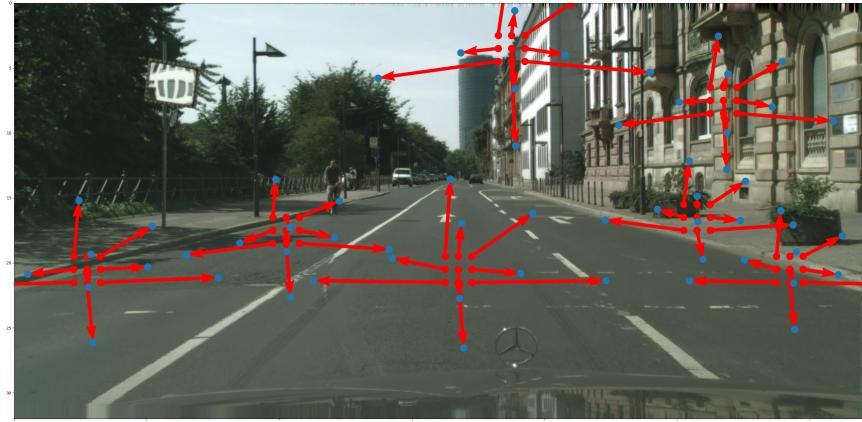


図 4.2: DHRNet の offset の可視化

可視化結果(図 4.2)の評価画素ごとに受容野の形状を比較すると大きさによる変化は確認できるが、形状の変化はあまり見られない。次に 4.3 と 4.4 を見ると、-6 以上 6 以下の範囲外に値が存在する。よって、DHRNet(deform-conv 1 層)を実験に使用するのに適しているといえる。

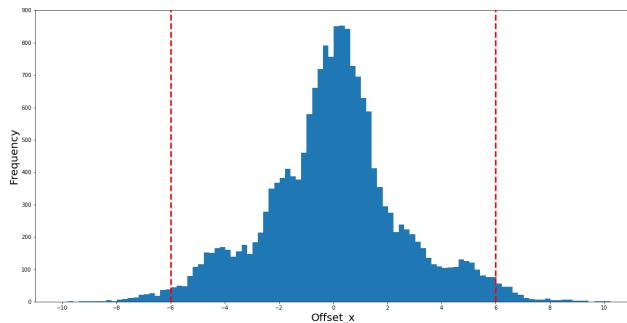


図 4.3: offset の x 成分の分布

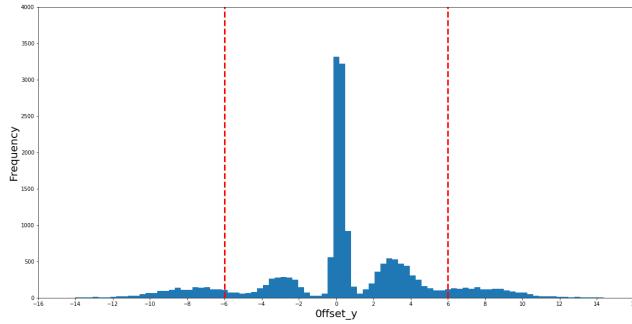


図 4.4: offset の y 成分の分布

#### 4.2.2 提案手法の有効性の確認

実験 1 で DHRNet(deform-conv 1 層) が評価実験のベースとできることを確認した。よって提案する活性化関数の評価を行う、さらにカーネルの中心 offset を固定することで cnn の位置普遍性を担保することの有効性の検証も行う。仮説としては、offset の範囲に制限を設けることにより、学習が促進され、中心の offset を 0 に制限することでさらに学習が進むと考えた。この仮説より、2 つの条件で実験を行った。

**提案手法 1** DHRNet(deform-conv 1 層) の offset 算出部分に hard tanh6 を導入

**提案手法 2** DHRNet(deform-conv 1 層) の offset 算出部分に hard tanh6 を導入することに加え、中心の offset ベクトルを 0 に固定

実験 2 のクラス全体の結果を表 4.4、クラスごと結果を表 4.5 に示す。

表 4.4: 実験 2 の結果 (クラス全体)

	baseline	MIoU(%)
	HRNet	70.26
deform-conv 1 層	DHRNet	73.52
deform-conv 1 層 + hard tanh6	DHRNet	<b>73.85</b>
deform-conv 1 層 + hard tanh6 (中心の offset 0)	DHRNet	73.28

表 4.5: 実験 2 の結果 (クラスごと)

クラス	IoU(%)			
	HRNet	deform-conv 1 層	deform-conv 1 層 + hard tanh6	deform-conv 1 層 + hard tanh6 (中心の offset 0)
road	97.64	97.88	<b>97.89</b>	97.84
sidewalk	82.21	83.00	<b>83.03</b>	82.80
building	90.90	91.62	<b>91.66</b>	91.63
wall	47.10	51.90	50.37	<b>54.53</b>
fence	53.05	55.40	57.56	<b>57.80</b>
pole	60.42	61.67	<b>61.92</b>	61.71
traffic light	63.87	63.17	<b>64.86</b>	64.45
traffic sign	74.33	<b>75.27</b>	75.06	74.82
vegetation	91.89	91.97	<b>92.11</b>	92.08
terrain	62.19	62.52	<b>63.15</b>	62.51
sky	93.74	<b>94.40</b>	94.30	94.26
person	78.58	79.00	<b>79.45</b>	78.83
rider	53.95	57.44	<b>58.11</b>	55.60
car	93.14	<b>93.84</b>	93.64	93.81
truck	50.54	<b>67.19</b>	61.18	65.63
bus	70.68	<b>80.93</b>	79.74	78.34
train	47.08	61.85	<b>67.30</b>	58.55
motorcycle	49.82	54.48	<b>57.57</b>	52.73
bicycle	73.73	73.57	74.20	<b>74.46</b>

表 4.4 より, DHRNet(deform-conv 1 層) に hard tanh6 を適応した提案手法 1 は, DHRNet(deform-conv 1 層) から 0.33pt 向上している. そして DHRNet(deform-conv 1 層) に hard tanh6 を適応し, 中心の offset を 0 に固定した提案手法 2 は, DHRNet(deform-conv 1 層) から 0.24pt 低下した. また 4.4 より, hard tanh6 を適応した提案手法 1 は, DHRNet(deform-conv 1 層) と比べて 19 クラス中 13 クラスで精度が高かった. よって, deformable convolution の offset の範囲に制限を設けることで学習が促進することがわかった. 次に offset に制限を加えたことでどのように受容野が変更されているかを確認するために, offset の可視化結果を示す.

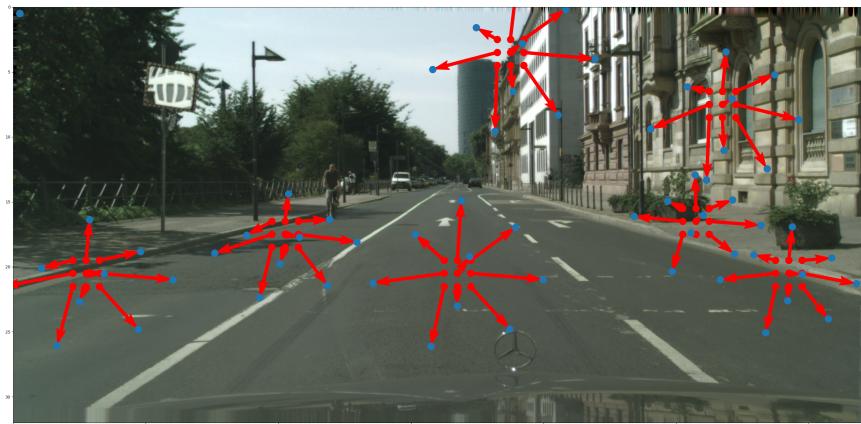


図 4.5: deform-conv+hard tanh6 の offset

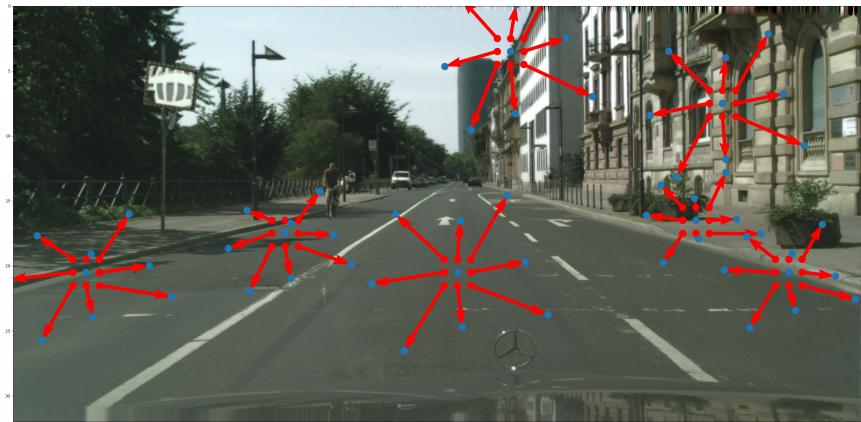


図 4.6: deform-conv+hard tanh6(中心の offset ベクトルを 0 に固定) の offset

図 4.5 と図 4.6 より、受容野が指定した範囲内に制限されていることが確認できた。

# 第5章

## 結論

本研究では受容やの大きさや形状が可変な deformable convolution が物体によって受容野の形状をどのように変更しているのかを受容やを可視化することにより視覚的に評価した。その結果、受容野が学習できる範囲外までの広がっていると考えた。そこで受容野の広がる範囲を offset を算出する特徴マップが表現できる範囲内に抑えるために新たな活性化関数を提案した。提案した活性化関数を MIoU, IoU で評価するのに加え、受容野を可視化して意味のある変形をしているかを確認した。結果は、MIoU が 0.3pt 向上し、IoU は多くのクラスにおいて精度の向上をパラメータの追加をせずに達成した。よって deformable convolution の受容野を制限することで学習が促進すると結論付けられる。

## 謝辞

本論文執筆にあたり、貴重な時間を割き熱心にご指導頂いた荒井秀一教授に心から感謝します。また自らの研究が忙しい中、貴重な時間を割いて助言をして下さった知識情報処理研究室の先輩を始め、1年間苦楽を共にした知識情報処理研究室の皆様に感謝します。

2022年年01月24日  
折田 汐凪

## 参考文献

- [1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.
- [2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.
- [3] Roberto Cipolla Jamie Shotton, Matthew Johnson. Semantic texton forests for image categorization and segmentation. In *Proc. Computer Vision and Pattern Recognition (CVPR 2008)*, pp. 1–8, 2008.
- [4] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, pp. 2–23, 2009.
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, Vol. 25, pp. 1097–1105. Curran Associates, Inc., 2012.
- [6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.
- [8] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2019.

- [9] Daiki Ando and Shuichi Arai. Semantic segmentation using hrnet with deform-conv for feature extraction dependent on object shape. In *2021 3rd International Conference on Cybernetics and Intelligent System (ICORIS)*, pp. 1–5, 2021.
- [10] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 39, No. 12, pp. 2481–2495, 2017.
- [11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.
- [12] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1925–1934, 2017.
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [14] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pp. 2847–2854. PMLR, 2017.
- [15] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.
- [16] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.
- [17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. p. pp. 37:448–456, 02 2015.
- [18] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2604–2613, 2019.
- [19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

- [20] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- [21] Geoffrey E Hinton and Russ R Salakhutdinov. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pp. 1607–1614, 2009.

# 付 錄 A

## 基礎事項

### A.1 活性化関数

活性化関数とは、線形分離できない問題を解くために、ニューラルネットワークの各ユニットからの出力を正規化する非線形な関数である。代表的な関数として Sigmoid 関数や Tanh 関数などがあるが、Semantic Segmentation のネットワークの場合、中間層では “ReLU”[20]、出力層では分類問題で多く用いられる “Softmax”[21] を使用するのが一般的である。以下に本研究で使用する活性化関数について記述する。

#### Softmax 関数

Softmax 関数 [21] は主に多クラス分類問題での出力層に用いられる。出力層のニューロンの数が  $n$  個あるとして、 $k$  番目の入力  $x_k$  に対する出力  $y_k$  を求める計算式は、

$$y_k = \frac{\exp(a_i)}{\sum_j^n \exp(a_j)}, \quad (i = 1, \dots, n) \quad (\text{A.1})$$

となる。Softmax 関数の出力は、0 から 1 の間の実数になる。また、Softmax 関数の出力の総和は 1 になる。この性質から Softmax 関数の出力は多クラス問題における確率値として解釈できる。

#### ReLU(Rectified Linear Unit) 関数

ReLU 関数 [20] は入力値が正ならば値をそのまま出力し、負ならば 0 を出力する関数である。入力を  $x$  としたときの式は、

$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases} \quad (\text{A.2})$$

となる。このように ReLU 関数はシンプルであることから、層の数が多くなっても安定した学習ができるため、深層学習の研究において最もよく用いられている活性化関数である。

## tanh(Hyperbolic Tangent) 関数

tanh 関数は式 ( A.3) に示す関数であり, 出力は-1 から 1 の範囲である.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (\text{A.3})$$

## hard tanh 関数

hard tanh 関数は式 ( A.4) に示す関数であり, tanh よりも計算量が少ない.

$$\text{hardtanh}(x) = \begin{cases} 1(x > 1) \\ x(-1 \leq x \leq 1) \\ 0(x < -1) \end{cases} \quad (\text{A.4})$$

## A.2 Batch Normalization

“Batch Normalization”[17] は, データ全体をいくつかの小さなデータの集合に分割した “ミニバッチ” ごとにデータの分布を平均を 0, 分散が 1 になるように正規化する手法である. この手法により, 学習データに過度に適応してしまう “過学習” を抑制することができる.

## A.3 バイリニア補間

バイリニア補間は画像の拡大, 回転, 変形を行うときの画素補間の一種である. 求めた い座標の画素値の周囲  $2 \times 2$  画素値を参照し, その加重平均値を用いて補間する. バイリニア補間の式を式 ( A.5) に示す. なお  $f$  は画素値,  $(x_0, y_0)$  は着目点の座標,  $(x, y)$  は  $(x_0, y_0)$  の周囲の格子点座標を表す.

$$\begin{aligned} f(x_0, y_0) &= f(x, y)(1 - \alpha)(1 - \beta) + f(x + 1, y)\alpha(1 - \beta) \\ &\quad + f(x, y + 1)(1 - \alpha)\beta + f(x + 1, y + 1)\alpha\beta \end{aligned} \quad (\text{A.5})$$

$$\begin{aligned} x &= \lfloor x_0 \rfloor \\ y &= \lfloor y_0 \rfloor \\ \alpha &= x - \lfloor x_0 \rfloor \\ \beta &= y - \lfloor y_0 \rfloor \end{aligned}$$

## A.4 $1 \times 1$ 置み込み (Pointwise Convolution)

$1 \times 1$  置み込み (Pointwise Convolution) は,  $1 \times 1$  カーネルで置み込み演算を行う. 通常の Convolution は入力特徴マップの空間方向(幅, 高さ)とチャンネル方向に同時に置み込む. 対し,  $1 \times 1$ Convolution では, チャンネル方向のみに置み込む. そのため  $1 \times 1$ Convolution はチャンネル数の調整に用いられる.

## A.5 誤差逆伝播法 (Back-propagation)

ニューラルネットワークは損失関数(付録 A.6)が最小になるように学習していくが, その際に損失関数が減少する方向を示す勾配が必要になる. この勾配を効率よく求める方法として誤差逆伝播法がある. 誤差逆伝播法は, 誤差を出力層から前の層へ次々と伝播していくことで, 各層の重みの勾配を求める.

## A.6 損失関数

損失関数はネットワークの出力結果と正解データの誤差を表す関数である. 出力結果が正解データに近づくほど損失関数から得られる誤差は小さく, 出力結果が正解データから離れるほど誤差は大きくなる. ニューラルネットワークでは損失関数が最小になるように学習する.