

卒業論文

題目

ニューラルネットワークを用いた手術画像
セグメンテーションシステムのFPGA実装

指導教員

柴田裕一郎 教授

2019年度

長崎大学工学部工学科 情報工学コース

藤田光暉 (35316034)

論文内容の要旨

情報工学コース

履修番号	35316034	氏名	藤田光暉
研究室名	柴田研究室		
研究題名	ニューラルネットワークを用いた手術画像 セグメンテーションシステムの FPGA 実装		

論文内容の要旨

あぶすと

目 次

第 1 章 緒論	1
第 2 章 背景と目的	2
2.1 関連研究	2
2.2 プロジェクトの現状	3
2.3 研究目的	4
第 3 章 理論	5
3.1 ニューラルネットワーク	5
3.2 畳み込みニューラルネットワーク	7
3.3 セマンティックセグメンテーション	8
3.4 FPGA	9
第 4 章 設計と実装	10
4.1 アルゴリズム	10
4.1.1 ネットワーク構成	10
4.1.2 畳み込み層	11
4.1.3 プーリング層	12
4.1.4 アンプーリング層	13
4.2 設計と実装	13
4.2.1 システム構成	14
4.2.2 stream_patch モジュール	14
4.2.3 ExtNet・RdcNet・ItgNet モジュール	15
4.2.4 pooling モジュール	17
4.2.5 unpooling モジュール	18
4.2.6 buf モジュール	20
4.3 学習	21
4.3.1 ツール	21
4.3.2 学習データ	21
4.3.3 量子化手法	21
第 5 章 評価と考察	23
5.1 評価	23
5.2 考察	23

図目次

2.1 U-Net ネットワーク構造 : [6] より引用	3
3.1 各ニューロンにおけるプロセスの模式図 ($n = 2$)	5
3.2 ニューラルネットワークの例	6
3.3 畳み込み層：畳み込み演算を「*」で表記	7
3.4 pooling (max pooling)	8
3.5 セグメンテーションの例 : [9] より引用	8
3.6 unpooling	9
4.1 ネットワーク全体図	10
4.2 3種の小規模ネットワーク	11
4.3 パディングの有無によるサイズ変化の違い	12
4.4 プーリング層の適用による縮小	13
4.5 アンプーリング層の適用による拡大	13
4.6 システム構成概略図	14
4.7 パッチ切り出し	15
4.8 layer モジュールによるネットワーク作成イメージ	16
4.9 加算器ツリーによる畳み込み演算	16
4.10 ツリーによる pooling	17
4.11 有効画素が出力されるクロック数の変化	17
4.12 有効画素の出力タイミングとバッファサイズの変化	18
4.13 状態遷移図	19
4.14 LEVEL ごとの状態遷移イメージ	19
4.15 buf モジュール	20
4.16 実際の手術画像データ	21
4.17 教師データ	21
5.1 Sample 図	23
6.1 Sample 図	24

表目次

4.1 拡張方向に基づく座標値の変更	20
5.1 Sample 表	23
6.1 Sample 表	24

第1章

緒論

近年、医療現場において腹腔鏡手術の需要が高まっている[1]。腹部に小さく開けた穴から内視鏡を挿入し対象部位の外科手術を行う腹腔鏡手術は、開腹手術に比べ出血量や術後の回復期間¹などにおいて患者への負担が少ない。しかし、腹腔鏡手術には通常の執刀医に加え、内視鏡操作を行う技師が必要となる。患者数の増加や医師の地域偏在に起因する医療従事者不足が医療業界の深刻な問題となっているが、内視鏡技師もその例外ではなく、拡大する腹腔鏡手術需要に対応しきれていないのが現状である。

一方で、医療分野におけるロボットによる支援の進歩は目覚ましく、その活躍は神経外科、腹腔外科、胸部外科など多岐に渡る。既に実績を残しているロボットとしては内視鏡下手術支援ロボットである「da Vinci(ダビンチ)[3]」が挙げられ、2018年にはダビンチを使って行われた手術が約100万件に達した。このような医療用ロボットは年々開発が進められている。

これらの状況を踏まえ、長崎大学工学部、長崎大学病院、中央大学工学部の共同研究として、胆囊摘出手術を想定した内視鏡操作ロボットの開発が行われている。このロボットは内視鏡のカメラワークを自動で調節し、執刀医のみによる腹腔鏡手術を可能にすることを目標としており、この実現によって拡大する手術需要に対応することが期待される。なお、胆囊摘出手術は腹腔鏡手術による施術が特に増加している手術の一種であり²、ダビンチの分野毎における利用数の推移[3]からもその需要の高まりが窺える。

胆囊周辺の特定部位へとカメラ画角を調整するには、前段階として内視鏡画面内の胆囊周辺部位座標を取得する必要がある。そこで本プロジェクトでは、畳み込みニューラルネットワークを用いた推論によってセグメンテーションを行い、リアルタイムでの判別が可能なシステムの設計を行っている[4]。本研究では、今後行われるであろうシステムの機能拡張に際してリアルタイム性を維持できるようにするために、ハードウェア化による高速化手法を提案し、既に実装されているシステムと、正答率・処理速度の観点から比較・評価を行う。

本論文の構成は以下の通りである。まず第2章において、本研究の背景と目的を述べる。第3章では、ニューラルネットワークを始めとした、本研究におけるシステムの基となる理論について説明を行う。第4章では、構築されるネットワークの特徴と、ハードウェアに実装するための手法について述べる。続く第5章では、FPGA上に実装されたシステムの評価・考察を行い、最後に第6章にて、本研究における結論を述べる。

¹良好ならば術後4日目に退院、仕事復帰は約1週間[2]

²29の病院を対象とした調査では胆囊摘出における腹腔鏡手術の割合は平均で93.3%[1]

第2章

背景と目的

内視鏡操作をロボットで行うにあたり、画角調節のため対象物体である胆嚢周辺部位の判別が必要となる。本プロジェクトでは、ニューラルネットワークを用いたセグメンテーションシステムによってこれを実現する。しかし、一般的にニューラルネットワークは学習に多量のデータセットを必要とするのに対し、医用画像はその特殊性¹からデータセットが少ないという問題点がある。

本章では、このような問題を抱える医用画像解析の分野において、機械学習を用いて一定の成果を納めた研究事例を挙げるとともに、実装済みのシステムとプロジェクトの現状、それらを踏まえた本論文における研究目的について述べる。

2.1 関連研究

U-Net[6] は、医用画像セグメンテーション用として提案され、2015 年の ISBI (IEEE International Symposium on Biomedical Imaging) で Dental X-Ray Image Segmentation Challenge と Cell Tracking Challenge の 2 部門で優勝するなど、高い評価を得ているセマンティックセグメンテーション手法である。画像のダウンサンプリング (低解像度化) に対して、アップサンプリング (高解像度化) を行うのは他のセグメンテーション手法でも見られるが、U-Net の特徴はアップサンプリング時にダウンサンプリング前のデータを連結することにある。この連結はスキップ接続と呼ばれ、データが伝搬する過程で詳細な情報が失われてしまい、セグメンテーション結果が粗くなる (粗大化する) のを防ぐ効果を持つ。図 2.1 にネットワーク構造を示す。ただし、U-Net はネットワークサイズが大きいことから、少ないデータセットでは訓練データを丸暗記してしまう過学習を引き起こす可能性が高い。

国立研究開発法人国立がん研究センターと日本電気株式会社は、深層学習を用い、大腸がん及び前がん病変を内視鏡検査時にリアルタイムに発見するシステム [7] の開発に成功した。このシステムは、内視鏡医師によってアノテーション (教師データとなるようにラベリング) された約 5000 例の内視鏡画像を教師データとして学習が行われる。約 5000 例という値は一般的な深層学習の教師データとしては少ないながらも、98 % という高い認識率と約 30fps (frames per second) の高速処理を実現している。

¹情報セキュリティやデータ提供に関する負担などの観点からガイドライン [5] に沿った適切な取得・運用が求められることに加え、教師データの作成は専門の医師による手作業で行われることが多い。

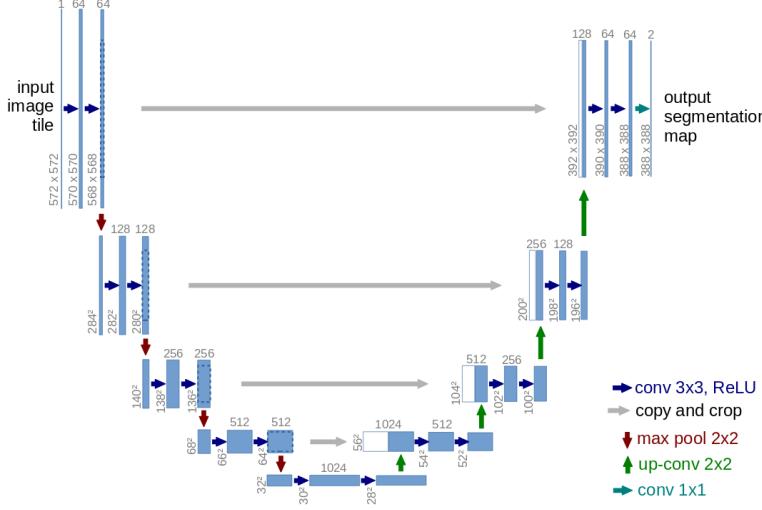


図 2.1: U-Net ネットワーク構造 : [6] より引用

2.2 プロジェクトの現状

本プロジェクトでは、GPU によるニューラルネットワークを用いたセグメンテーションシステムが実装されており、評価用データを用いた実験では正答率 61.2 %、約 17fps の性能を達成した [4]。

このシステムの検証実験としてブタの生体を用いた動作テストを行った。ブタは形態学的、生理学的にヒトとの類似性が高いことが知られており、医学、免疫学、再生医療などの分野において広範囲に利用されている。検証実験の結果、セグメンテーションシステム自体の精度向上の他に、以下のような機能の必要性が見出された。

- ブタとヒトの差異にシステムを対応させるための前処理
- 手術補助としてのセグメンテーション結果のオーバーレイ

ブタはヒトとの類似性が比較的高いものの、臓器表面の色彩やテクスチャにおいて多少の差異が存在するため、ヒトのデータセットによって学習された当システムの検証対象としては適切ではない可能性がある。しかしながら、新たにブタのデータセットを十分に用意するのは難しく、たとえ用意できたとしてもそれはブタに適応したシステムとなってしまう。そこで、カメラからの動画像にブタの臓器画像をヒトのそれに近づける前処理を追加する予定である。これにより、ブタを用いた検証実験によって当システムのヒトへの有効性を正しく評価することを狙う。後者は執刀医に対する手術補助を意図して、内視鏡映像にセグメンテーション結果をオーバーレイするものである。

このような操作が現在のシステムに追加された場合、その分の処理時間が増加し、現在の実装ではリアルタイム性が失われる可能性がある。そこで、実装されているセグメンテーションシステムをより高速にし、かつ処理の追加にも適した実装とすることで、当システムのリアルタイム性を維持できるようにする必要がある。

2.3 研究目的

本研究は、現在 GPU を用いて実装されている胆嚢周辺画像のセマンティックセグメンテーションの高速化手法として、FPGA によるシステムのハードウェア化を提案し、それによる演算速度の変化を検証することを目的とする。

畳み込みニューラルネットワークは、推論、学習ともに並列性の高い大量の積和演算を行うことから、一般的には GPU による処理が行われることが多い。しかしながら、動画像を対象とした畳み込み処理においては、単純な並列化に加えてパイプライン化による恩恵が大きく、計算分野によっては、FPGA 等によるハードウェア処理のほうが GPU よりも高速に動作することが知られている [8]。処理の追加に関しても、パイプラインで処理できるものならば、レイテンシの増加のみでスループットを維持した自然な実装が可能であり、画像処理の追加が予定される当システムに適した手法であるといえる。

また、FPGA は電力効率の観点から組み込み系機器での運用に適しており、持ち運びのできる組み込み系機器での運用が想定される当システムにおいては大きな利点となる。そこで本研究では、広範な機器で利用でき、電力効率にも優れたシステムを構築可能でありながら、より高速な動作が見込める FPGA を利用する。

第3章

理論

本章では、システムの実装に用いる理論についての説明を行う。

3.1 ニューラルネットワーク

ニューラルネットワーク (Neural Network) は、生物の脳内におけるニューロン (神経細胞) の結びつき方をモデルにした情報処理システムである。学習能力を持つため、サンプルとなるデータに基づき必要とされる機能を自動形成することができる。

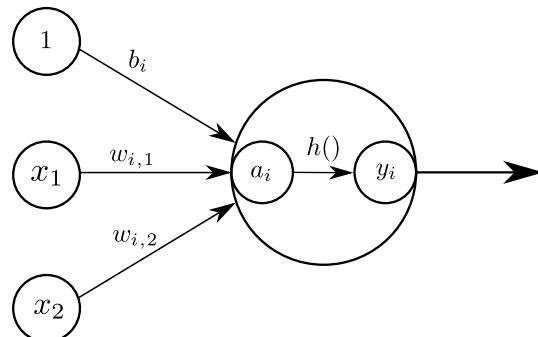


図 3.1: 各ニューロンにおけるプロセスの模式図 ($n = 2$)

ニューラルネットワークを構成するニューロンの模式図を、前段のニューロン数 n が 2 のときを例として図 3.1 に示す。ニューロン i の出力 y_i は、入力 x_j ($1 \leq j \leq n$) を基に、3.1 式により決定される。

$$y_i = h(b_i + \sum_{j=1}^n x_j w_{i,j}) \quad (3.1)$$

各入力 x_j には固有の値である重み $w_{i,j}$ が設定され、同じく固有の値であるバイアス b_i と共に計算に用いられる。この固有の値がニューラルネットワークの機能を決定づける要素であり、適切な値に設定することで必要とする機能を形成する。また、 h は活性化関数と呼ばれる関数であり、以下のような非線形関数が用いられることが多い。

$$h(x) = (1 + e^{-x})^{-1} \quad (\text{標準シグモイド関数}) \quad (3.2)$$

$$h(x) = \max(0, x) \quad (\text{ReLU}) \quad (3.3)$$

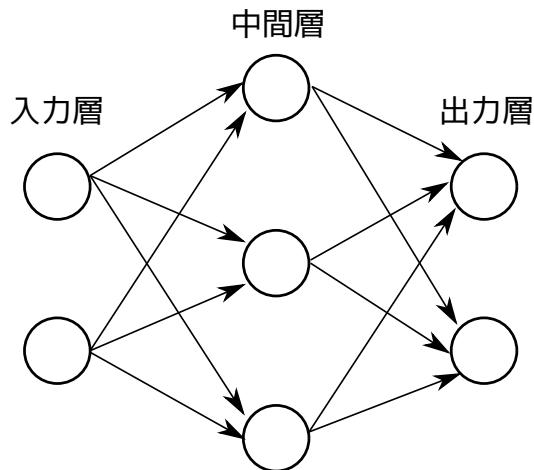


図 3.2: ニューラルネットワークの例

ニューラルネットワークの代表的な構造として、順伝播型ニューラルネットワークの1種である多層パーセプトロンを例に挙げる。中間層が1層以上存在する多層パーセプトロンでは、任意の連続関数を近似可能であることが知られている。ただし、線形な関数のみで構成されるネットワークは、どんなに層を増やしたとしてもそれと等価な单層ネットワークが存在するため、層を増やす恩恵を得るには非線形な関数が各層で用いられる必要がある。そのため、一般的に活性化関数には非線形関数が用いられている。

先述した通り、ニューラルネットワークで期待される機能を実現するには、各ニューロンの重みとバイアスを適切に設定しなければならない。この調整をデータに基づいて自動で行う仕組みが学習である。学習には大きく分けて教師あり学習と教師なし学習があるが、ここでは教師あり学習について説明を行う。

教師あり学習では、入力データと教師データが対になって与えられたデータセットを利用し、ネットワークに入力データを渡した際の出力が教師データと一致するように重みやバイアスを変化させていく。一般的に、出力精度の指標としては2乗和誤差や交差エントロピー誤差などの損失関数が、学習手法としては誤差逆伝播法が用いられる。入力データに基づく出力を y_k 、それと対応する教師データを t_k として以下に損失関数の例を示す。

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2 \quad (\text{2乗和誤差}) \quad (3.4)$$

$$E = - \sum_k t_k \log y_k \quad (\text{交差エントロピー誤差}) \quad (3.5)$$

損失関数とは、出力精度の低さを示す指標であり、これによって得られた誤差を最小とするように重みおよびバイアスを変更する。また、重みに基づいて誤差を逆伝播させ、中間層の重みおよびバイアスを順次更新していく。損失関数は逆伝播可能な関数ではなくてはならず、活性化関数との組み合わせによっては学習が遅くなることもある。上記の損失関数は逆伝播可能で、一般低な活性化関数との利用に適しており、最尤推定に則った損失関数であるため用いられることが多い。

3.2 署み込みニューラルネットワーク

署み込みニューラルネットワーク (Convolutional Neural Network : CNN) は、図 3.2 に示したような、隣接する層の全ニューロン間で結合がある (全結合) ニューラルネットワークとは異なり、入力と設定されたフィルタの署み込み演算に基づいて出力の決定を行うニューラルネットワークの一種である。画像認識や音声認識など CNN の活用形態は多岐にわたり、本研究においても画像認識を目的として CNN を利用している。本項では、本研究で行われる 2 次元画像に CNN を適用する場合を例として CNN について説明を行う。

画像認識における全結合ネットワークには、空間的情報が失われるという問題点がある。入力が 2 次元画像の場合、空間的に近いピクセルは似たような値である、距離の離れたピクセルは互いに影響を及ぼさない、画素の RGB 値には密接な関連があるなどといった、空間的形状には汲み取るべき本質的なパターンが含まれていると考えられる。しかし全結合層はこのような形状を無視し、すべて同等のニューロンとして処理を行うため、空間的情報を生かすことができない。

そこで CNN は形状を維持するために署み込み層を利用する。署み込み層で行われる署み込み演算は $n \times n$ サイズの入力画像を $X(i, j)$ 、 $m \times m$ サイズのフィルタを $F(i, j)$ として、式 3.6 のように定義できる。

$$(X * F)(i, j) = \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} X(i+k, j+l)F(k, l) \quad (3.6)$$

また、 $n = 4, m = 3$ として、署み込み層全体の処理を図 3.3 に示す。

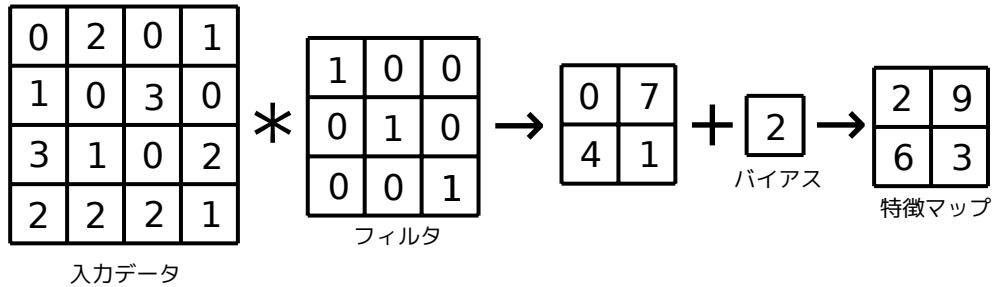


図 3.3: 署み込み層：署み込み演算を「*」で表記

以上のように、フィルタと入力データの対応する要素の積和演算を行い、対応する場所へ格納していく。このフィルタこそが重みに対応するパラメータであり、フィルタの値が CNN の動作を決定付ける。バイアスは全結合ニューラルネットワークと同様に、フィルタ (重み) の適用後に加算される。

図 3.3 でも示されるように、フィルタの適用により出力 (特徴マップ) は入力データに比べて一回り小さくなる。これを回避するために、入力データの端に 0 を追加する (ゼロパディング) 操作を加える場合がある。これにより、図 3.3 のように、フィルタの大きさが 3×3 かつ、フィルタの適用範囲を動かす量 (ストライド) が 1 ならば、入力と出力の大きさを一致させることができる。フィルタやストライドの大きさが変化する場合は、それを考慮してパディングの大きさも変化させる必要がある。

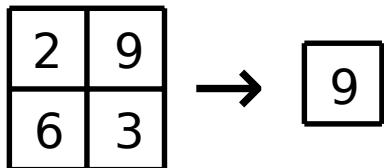


図 3.4: pooling (max pooling)

CNN を利用したアプリケーションでは、位置不变性（パターンの位置がずれても影響を受け辛い性質）が求められることが多く、その際にはプーリング層と呼ばれる新たな層がネットワークに追加される。画像を一定領域に区切り、各領域を最大値や平均値を用いて 1 画素に置き換える処理（pooling）により、パターンのずれを吸収させ、位置不变性を高める。その処理内容からわかる通り、プーリング層において学習するパラメータはなく、層間における画素ごとのニューロン数（チャネル）の変化もない。ただし、縦横方向の空間は小さくなるため、画素ごとの位置情報は失われる。よって、元々の位置情報が必要となるアプリケーションなどでは注意が必要となる。

3.3 セマンティックセグメンテーション

一般的な CNN による物体認識は、画像全体に対して何らかの判定・分類を行うものが多い。例としては、写真群から猫が写っている画像だけを抽出する機能などが挙げられる。一方で、セマンティックセグメンテーションは画素ごとに判定・分類を行う。例えば、複数の物体が写っている写真に対して、猫が写っている部分だけ塗りつぶしを行う機能が挙げられる。図 3.5 に示すセグメンテーション例では、猫とソファが区別されて塗り分けられている様子が確認できる。



図 3.5: セグメンテーションの例 : [9] より引用

1 画素だけを見てその画素が何にあたるか判定するのは通常不可能であるため、周辺画素ひいては画像全体を見て判定する必要がある。そこで、CNN による空間的情報を利用した判定を行うのだが、プーリング層を適用する場合、画素の位置情報が破棄されるという問題が発生する。

医用画像セグメンテーションを目的としたネットワークである U-Net[6] では、pooling によって縮小された特徴マップを後段で拡張することによって位置情報の復元を行う。この操作は pooling に対して unpooling（アンプーリング）と呼称される。U-Net における unpooling 処理を図 3.6 に示す。pooling の際に最大値だった画素の位置を保持し、unpooling では

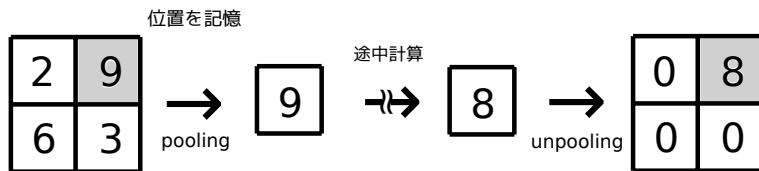


図 3.6: unpooling

保持された位置を利用して拡張を行う。また、U-Net はプーリング層に通す前の特徴マップを後段に直接連結させることで、粗大化を避けられないアンプーリング層の出力を補佐し、セグメンテーションの精度を高めている。

3.4 FPGA

Field Programmable Gate Array (FPGA) は、製造後に設計者が構成を設定できる集積回路である。FPGA は、複数入力のルックアップテーブル (LUT) 等で構成された論理ブロックを多数搭載し、LUT を書き換えることによって論理積や論理和といった様々な論理を表現できる。また、論理ブロック間を結ぶ内部配線についても構成を変化させることができるために、設計者は論理を表現したブロックを適切に組み合わせることによって任意の論理回路を実装する。

FPGA の設計は、一般的に Verilog HDL や VHDL といったハードウェア記述言語で行われる。用途に合わせて設計される集積回路である ASIC (Application Specific Integrated Circuit) に比べ、集積密度や電力効率、動作速度では劣る一方、開発・製造期間は短く、設計の変更も容易である。また、コスト面に関しても、製造のための初期コストは不要であり、FPGA 自体は汎用品であることから、少量生産においては生産コストでも有利である。

一般的に、ソフトウェアで動画像処理をするときは、動作クロック周波数の高い高性能 CPU が必要となる。一方ハードウェアは、回路を並列化やパイプライン化することで処理性能を上げることができ、ソフトウェアと比較して低い動作クロックで同等の処理を実現できる。そのため、画像処理システムには FPGA を始めとするハードウェアが用いられることが多い。また、計算分野によっては処理の並列化特性から高性能な CPU・GPU よりも高速に動作する可能性がある [8]。CNN もチャネル方向の並列化に加え、画素情報を順次走査で受け取りながらのストリーム処理が可能な点からハードウェアに適した計算処理であるといえる。

第4章

設計と実装

本章では、当プロジェクトにおいてソフトウェア実装されているシステムのネットワーク構造について説明し、そのハードウェア実装に際しての設計・実装方法について述べる。

4.1 アルゴリズム

本システムは、入力された手術画像に対して、ニューラルネットワークを用いたセグメンテーションを行い、ラベル画像を出力する。入力はサイズ 672×528 の RGB 画像であり、出力として 4 種類のクラスラベルを返す。クラスは、背景・胆嚢・胆嚢管・総胆管とする。

4.1.1 ネットワーク構成

本プロジェクトで実装されているネットワークの全体図を図 4.1 に示す。

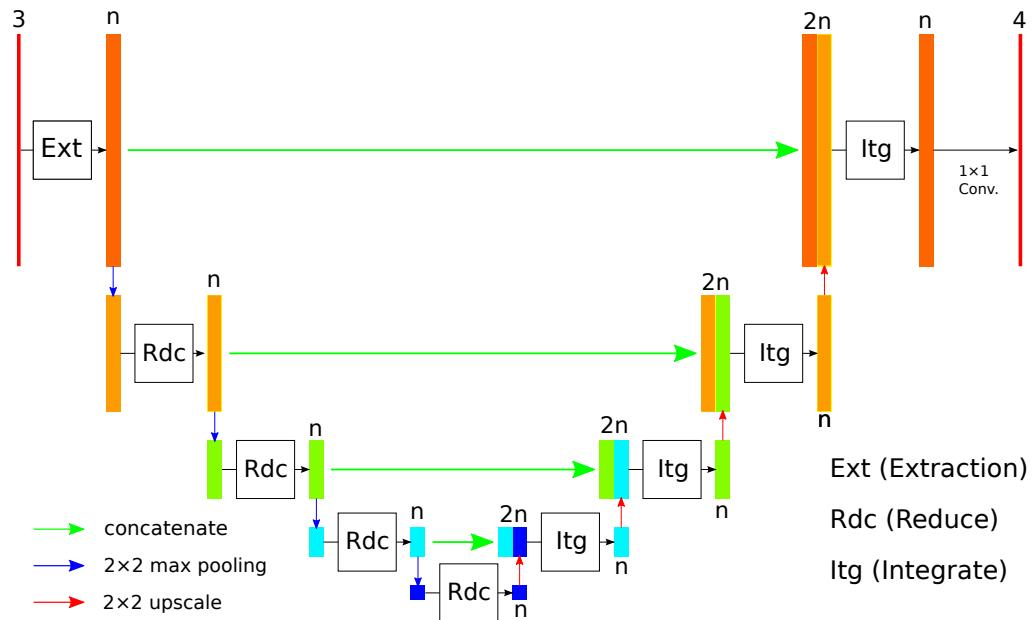


図 4.1: ネットワーク全体図

図 4.1 に則して説明を行う。まず入力画像は Ext という小規模ネットワークに与えられ、特徴マップが生成される。Ext は入力が RGB の 3 チャネル、出力が n チャネルとなっており、 n はパラメータで任意に設定できる。続いて、特徴マップは 2×2 max pooling で縮小された後、Rdc という小規模ネットワークに渡される。Rdc は入出力ともに n チャネルである。pooling による縮小と Rdc ネットワークの適用を再帰的に繰り返すことで、低次元への特徴マップの集約を行い、大域の情報を利用できるようにする。最下層に到達した後、unpooling による拡張が行われ、縮小処理を経ていない特徴マップと共に Itg という小規模ネットワークに渡される。よって Itg の入力は $2n$ チャネル、出力は n チャネルである。unpooling による拡張と前段の特徴マップの統合によって、プーリング層で失われた位置情報の復元を狙う。この拡張と Itg による統合は、縮小と Rdc の適用と同回数行われる。最後に、元画像と同じ大きさとなった特徴マップに 1×1 畳み込み演算を行い、指定の 4 クラスに対する尤度マップを生成する。

以下の項目では、ここに挙げた各処理について詳細を述べる。

4.1.2 畳み込み層

本ネットワークの畳み込み層は 3 つの小規模ネットワークから構成されている。図 4.2 にそれぞれの形状を示す。

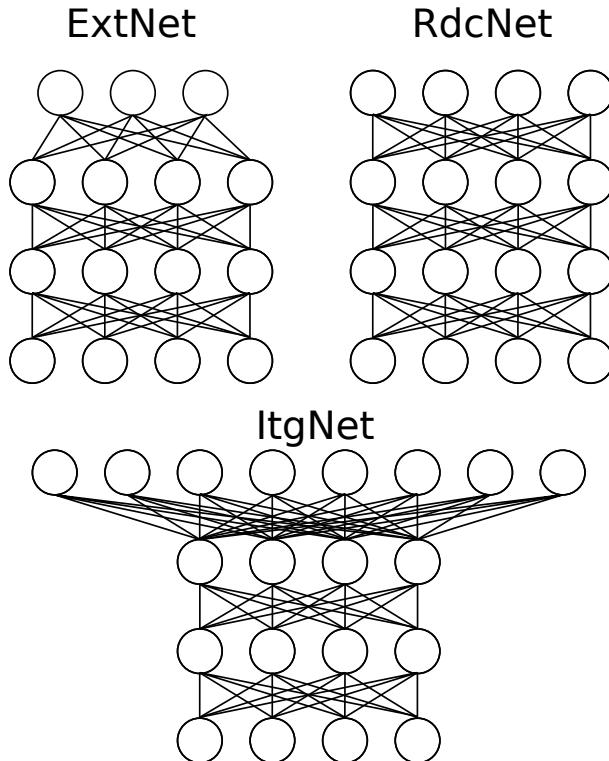


図 4.2: 3 種の小規模ネットワーク

フィルタサイズはいずれも 3×3 であり、入力と出力のサイズを同じにするため各層でパディングが行われている。活性化関数には、式 4.1 に示す Leaky ReLU[10] を用いる。

Leaky ReLU を用いる理由だが、式 3.3 に示した ReLU はいかなる負の入力に対しても 0 を出力するため、学習が進むにつれて絶対値の大きい負数が出力されることがある。本研究ではハードウェア化にあたり固定小数点演算を採用していることから、この現象は演算に必要なビット幅の増大や演算精度の低下につながる。一方 Leaky ReLU は、負領域においても 0 でない重み a を持つため、この問題を緩和できる。 a の値については、ビットシフト演算による単純な実装が可能な $a = 0.25 = 2^{-2}$ を採用した。

$$f(x) = \max(ax, x), a = 0.25 \quad (\text{Leaky ReLU}) \quad (4.1)$$

本ネットワークと U-Net の差異として、小規模ネットワークの再帰的な適用が挙げられる。医用画像はデータセットが乏しいため、訓練データを丸暗記してしまう過学習を引き起こす可能性が高い。そこで、小規模ネットワークの組み合わせによってネットワークを構築することで、記憶できるパラメータ数を意図的に減らし過学習を抑制する。加えて、異なる解像度のデータに対するネットワークの再帰的な適用により、サイズに左右されない、より普遍的な特徴抽出が期待できる点も、過学習抑制に繋がる。

また、各層でパディングが行われるのも U-Net との差異として挙げられる。これは、入力と出力のサイズを変化させないことで、設計を単純化できるためである。パディングを行わない場合、図 4.3 に示すように pooling · unpooling における動作が複雑化することが予測できる。各層でのパディングにより、小規模ネットワーク適用後、pooling では入力を半分に縮小し、unpooling では入力を 2 倍に拡張するという単純な動作となるため、設計を単純化できるという利点がある。

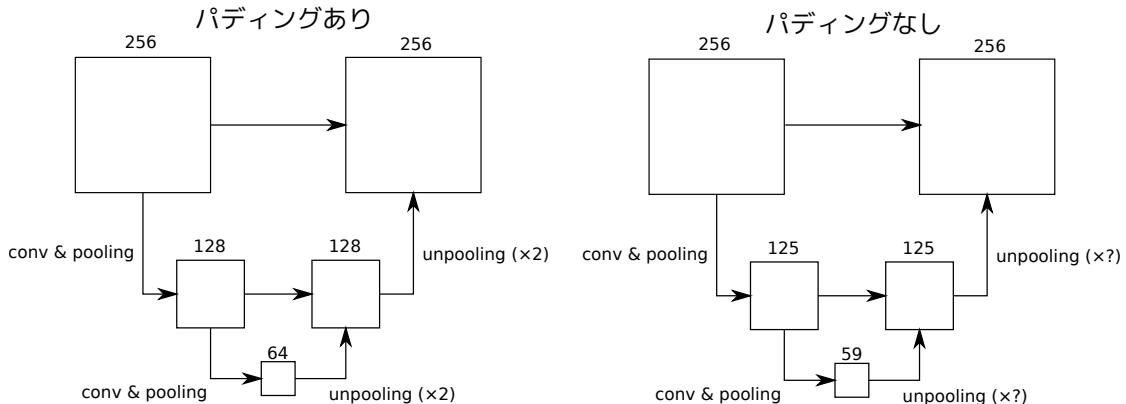


図 4.3: パディングの有無によるサイズ変化の違い

4.1.3 プーリング層

本システムのプーリング層では、一般的な CNN と同様に 2×2 max pooling を適用する。入力は 2×2 の領域に区切られ、各領域内の最大値 1 画素が出力される。よって、プーリング層の出力は前段からの入力に対し、縦横ともに半分のサイズとなる。図 4.4 にその様子を示す。4.1.2 節で述べたように、プーリング層の前後で実行される畳み込み層においてサイズは変化しないため、プーリング層でのサイズ変更は単純かつ規則的であることが確認できる。

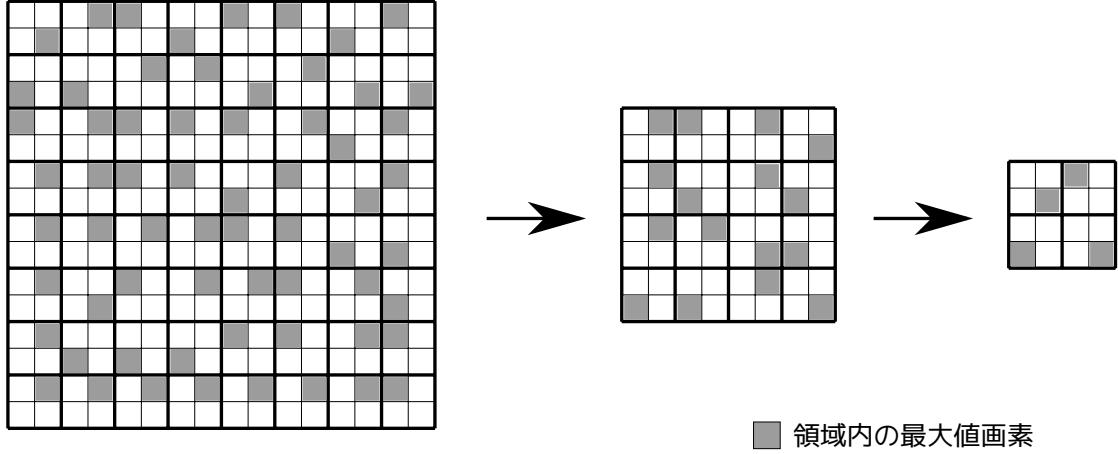


図 4.4: プーリング層の適用による縮小

4.1.4 アンプーリング層

本システムのアンプーリング層で行われている処理は図 3.6 とは異なり、値を周辺画素に転写する処理となっているため、位置情報を記憶する必要はない。プーリング層を経ていない特徴マップとサイズを一致させるには、4.1.2 節で述べたように縦横のサイズを 2 倍にすればよい。また、4.1.1 節で述べたように、アンプーリング層の適用はプーリング層の適用と同回数行われ、最終的な出力のサイズは最初の入力と一致する。アンプーリング層の動作を図 4.5 に示す。

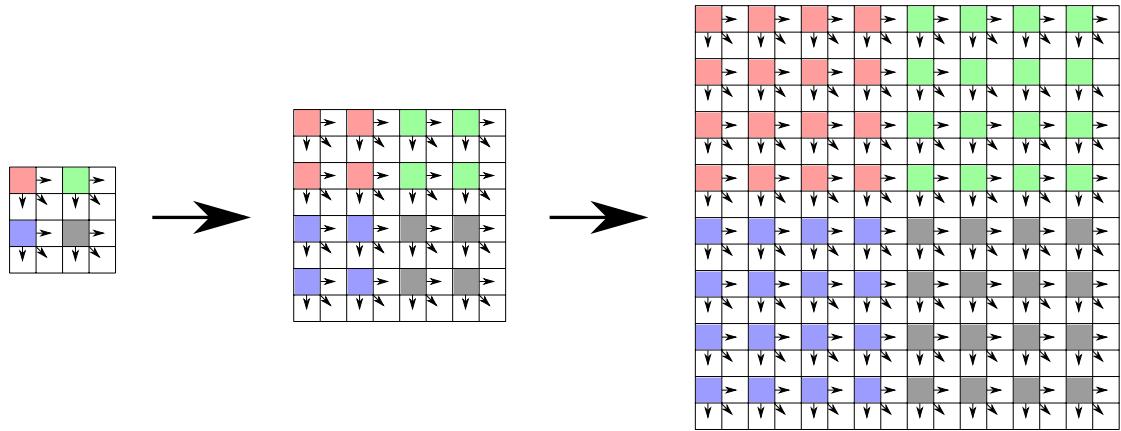


図 4.5: アンプーリング層の適用による拡大

4.2 設計と実装

4.1 節で説明したシステムのハードウェア化にあたり、外部からの入力画像は、順次走査により水平および垂直座標 (h_cnt, v_cnt) と共に画素値 (in_pixel) のストリームとして与えられることを想定する。それに伴い、出力は座標と共に 4 種類のクラスラベルが 1 画素ずつ返される。

FPGAへの実装は、Verilog HDLを用いたRTL記述にて行った。論理合成・配置配線にはVivado 2018.3を用い、ターゲットFPGAはVirtex UltraScale xcvu095-ffva2104-2-e-es2とした。

4.2.1 システム構成

図4.1のネットワーク構成を基に、図4.6のような設計を行った。

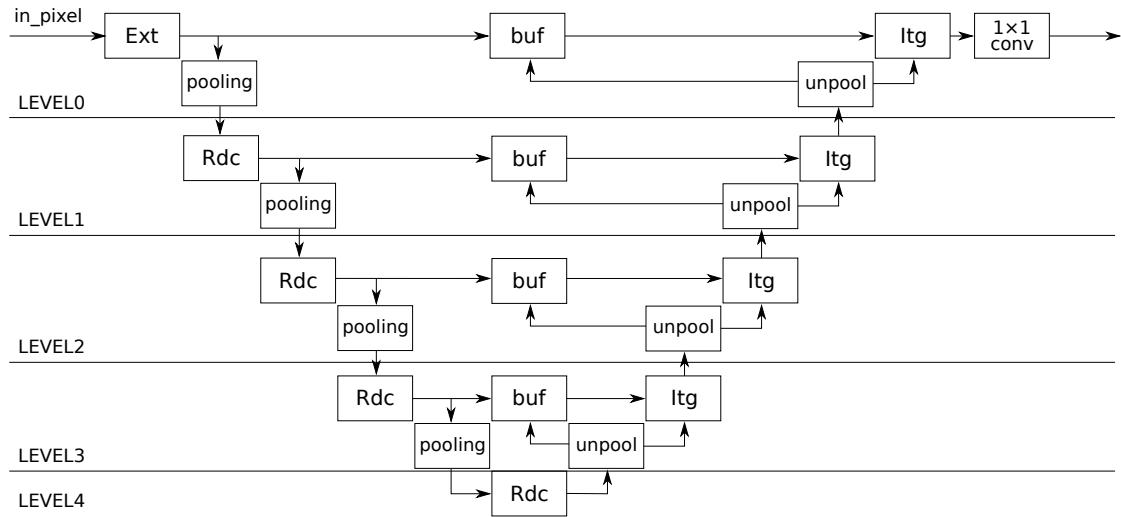


図4.6: システム構成概略図

システム全体は、順次走査により与えられる画像情報に対して、完全ストリーム処理を行うことができる設計となっている。畳み込み等のフィルタ演算は適切なバッファリングにより、フィルタと対応する画素の周囲画像(パッチ)に切り出され、ツリー構造の演算器で処理される。pooling・unpoolingにより画像サイズが変化しても、イネーブル制御によってストリーム処理を維持する。以下の項目では、ここに挙げた各処理を行うモジュールの詳細を述べる。

4.2.2 stream_patch モジュール

フィルタ演算が行われる各モジュール内では、当研究室内で利用されているパッチ切り出しのテンプレートモジュール(stream_patch)が用いられている。有効画素の入力ごとに、フィルタに対応した大きさのパッチが切り出される様子を図4.7に示す。これにより、有効画素が入力されるクロックごとのフィルタ演算が可能となる。

詳細は4.2.4・4.2.5節にて述べるが、本システムではpooling・unpoolingによって有効画素のタイミングが変化する。そこで、stream_patchにイネーブル信号を追加し、有効画素が入らないクロックではモジュールを停止させることで、全体のストリーム処理を維持する設計とした。ストリームパッチモジュール内で用いられているバッファ用のメモリは、Vivadoに組み込まれたIPカタログにより生成されたシンプルデュアルポートメモリであ

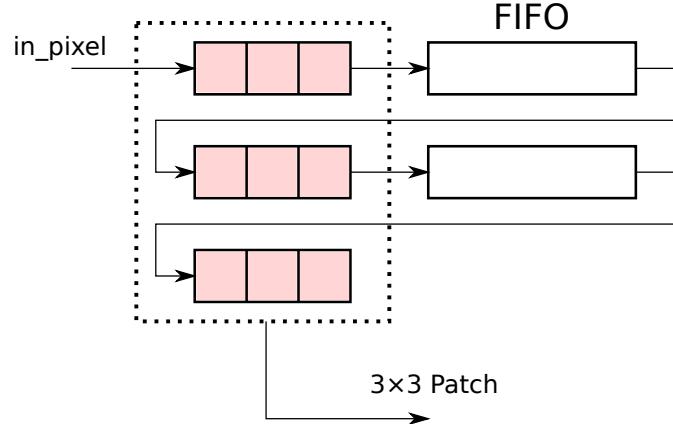


図 4.7: パッチ切り出し

る。このメモリは書き込みイネーブルを入力に持つため、0 を入力することで図 4.7 に示した FIFO を停止させることができる。

`stream_patch` の変更により、フィルタ演算を行うモジュールは適切なイネーブル信号を前段から受けとることで、サイズの変更による有効画素タイミングの変化に対応できるようになった。その他の停止操作や、各モジュールのイネーブル信号の出力方法については、以下の項目内にて詳細を述べる。

4.2.3 ExtNet・RdcNet・ItgNet モジュール

4.1.2 節で説明した通り、本システムの畳み込み層は 3 種類の小規模ネットワークから構成されるため、それぞれ 3 種類のモジュールとして実装を行うが、3 種類のネットワークには積和演算や活性化関数の適用など共通する部分も多い。そこで、図 3.1 に表したような、各ニューロンの計算を行う部分を層 (layer) としてとらえ、共通モジュールとして実装することで設計の単純化を狙う。図 4.8 に layer モジュールによるネットワーク作成のイメージ図を示す。layer モジュールはパラメータによって適用するニューロンの数を変更することができ、layer を重ねることで、多段ネットワークを実装する。よって、ネットワーク全体の入出力、layer モジュールを呼び出す段数、各 layer モジュールのパラメータの変更を行うことで、今回必要とするネットワークモジュールが実装可能となる。

layer モジュールについて説明する。layer モジュールは n チャネルの画素を 1 画素ずつ受け取り、`stream_patch` モジュールによってパッチに切り出した後、畳み込み演算、バイアス加算、活性化関数の適用を行う。畳み込みにおける加算器ツリーを図 4.9 に表す。

4.2.2 節で述べたように、各 Net モジュールは前段からのイネーブル信号を `stream_patch` モジュールに渡すことで、モジュールを停止させることができる。また、受け取ったイネーブル信号を自身のレイテンシ分遅延させて出力することで、後段のモジュールに対するイネーブル信号とする。画素の座標値 (h_cnt, v_cnt) に関しても、イネーブル信号と同様の遅延が行われて出力される。

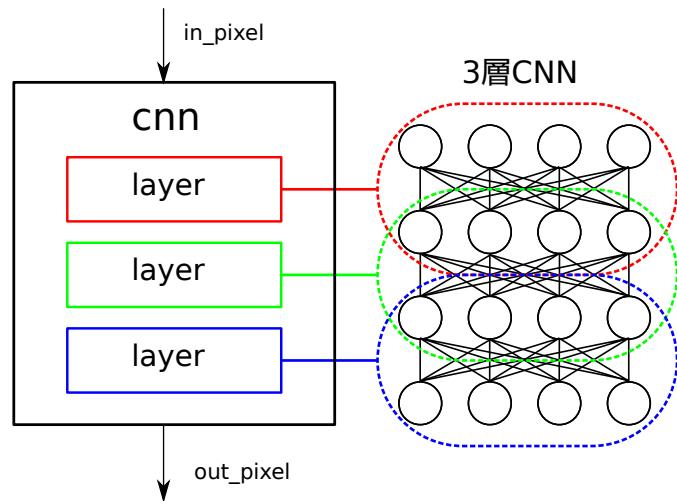


図 4.8: layer モジュールによるネットワーク作成イメージ

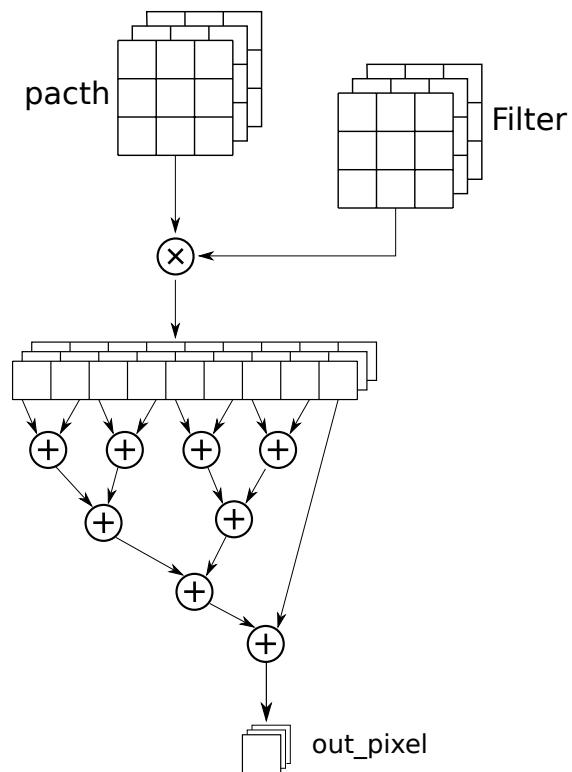


図 4.9: 加算器ツリーによる畠み込み演算

4.2.4 pooling モジュール

pooling モジュールも各 Net モジュールと同じく、stream_patch モジュールを用いて対象画素の入力ごとに動作することができる。ここでいう対象画素とは、今回 2×2 max pooling を適用するため、画像全体を 2×2 の領域で区切った際の右下画素のことをいう。右下画素が入ったクロックで pooling を行い、イネーブル信号と合わせて出力を行う。切り出されたパッチ内の最大値を求めるツリーを図 4.10 に示す。

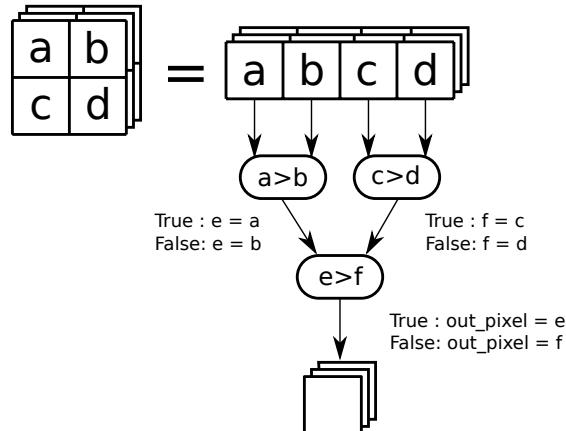


図 4.10: ツリーによる pooling

右下画素であることは、座標値の下位 1 ビットを用いた以下の条件文で判定できる。

- if (h_cnt[0] == 1 && v_cnt[0] == 1)

また、座標値 (h_cnt, v_cnt) の出力はサイズの変化に合わせて適切に変更される必要がある。4.1.3 節で述べた通り、各 pooling モジュールの出力サイズは入力に対して縦横それぞれ半分となるので、下位 1 ビットを切り捨てればよい。この画像サイズ縮小に伴い、後段のモジュールにおける有効画素が入るタイミングが変化する。図 4.11 にその様子を示す。左図は毎クロック有効画素を出力しているのに対し、pooling モジュールの適用ごとに有効画素を出力するクロックが減少していることがわかる。

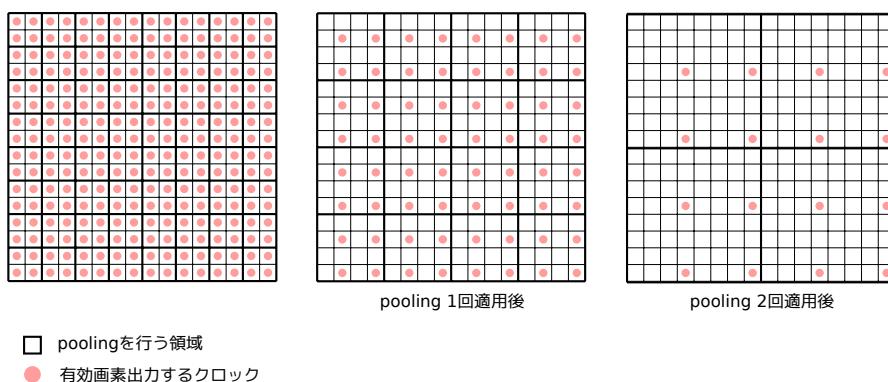


図 4.11: 有効画素が出力されるクロック数の変化

4.2.5 unpooling モジュール

4.1.4 節で述べたように、本システムにおける unpooling 動作は単純な画素拡張操作であるため、入力画素のバッファリングによって実装する。前段からの有効画素に対して、図 4.5 のように右上、左下、右下に拡張するにあたり、バッファの大きさを適切に決定する必要があるが、図 4.12 に示すように、unpooling の適用回数による有効出力タイミングの変化は規則的であるため、適用回数を基としたパラメータによってバッファの大きさを決定することとした。このパラメータを LEVEL とし、元々の入力画像と同じサイズに変化する際、つまり毎クロック有効画素を出力する unpooling の LEVEL を 0、以降を 1...2... と設定する。LEVEL に基づいて決定されるバッファの大きさを以下に示す。これにより、unpooling モジュールはパラメータとして LEVEL を設定するだけで再帰的な利用が可能な設計となった。

- 右上 : UppR_BUF = 2^{LEVEL}
- 左下 : LowL_BUF = $2^{\text{LEVEL}} \times \text{WIDTH}$ (入力画像横幅)
- 右下 : LowR_BUF = UppR_BUF + LowL_BUF

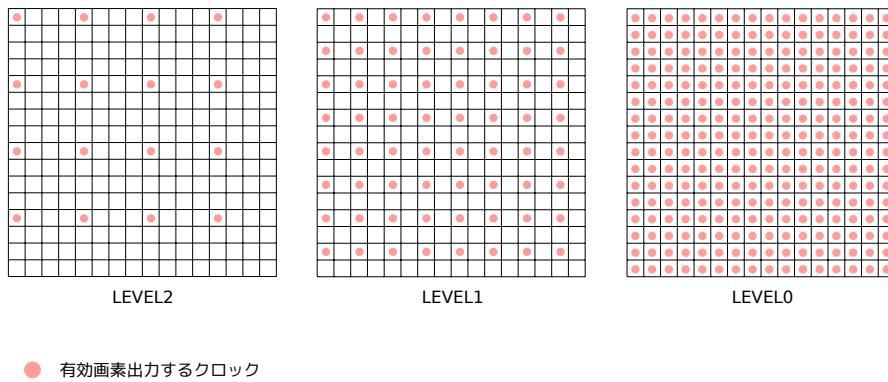


図 4.12: 有効画素の出力タイミングとバッファサイズの変化

画素の出力順は 4 方向の循環ではなく、行ごとに動作が有効画素が入力される行では左上・右上画素が、 $2^{\text{LEVEL}} \times \text{WIDTH}$ 後の行では左下・右下画素が出力される。そこで、前段からのイネーブル信号、バッファ分のカウント、行判定によって遷移するステートマシンとしての設計を行った。図 4.13 に状態遷移図を示す。

イネーブル信号で遷移する待機状態を初期状態とする。外部からのイネーブル信号によって左上に遷移。バッファ用のメモリに画素値を渡し、左上の出力をを行う。その後、カウンタによって LEVEL に基づくバッファ分待機し、右上に遷移する。右上はメモリから画素値を受けとり出力。同じくカウンタによってバッファ分待機し、左上に再び遷移する。これを 1 行分繰り返す。なお、行の終了も出力間の待機と同じようにカウンタを用いて判定している。1 行終了後、出力行判定で遷移する待機状態へと遷移する。ここで用いる出力行判定とは、左下・右下画素の出力を開始する行を判定することを指し、判定には LEVEL を用いた。左上・右上画素の出力行を 0 行目とした場合、左下・右下画素の出力行は 2^{LEVEL} 行目となる。そこで、1 行終了が判定された回数を用いて、出力行判定を行つ

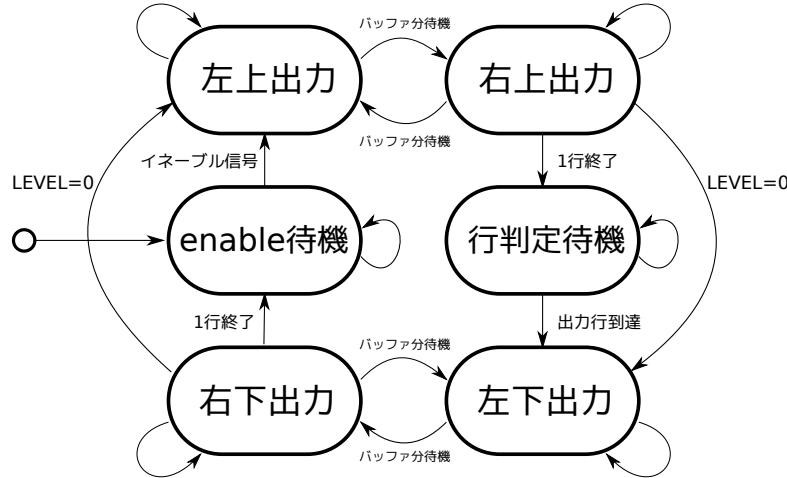


図 4.13: 状態遷移図

た。その後、左下・右下も左上・右上と同様に、遷移を繰り返しながら 1 行分の出力をを行い、1 行終了後イネーブル信号によって遷移する待機状態に戻る。図 4.14 に LEVEL ごとの状態遷移イメージを示す。外部からの入力のタイミングで $in_vcnt[0]$ が変化しているが、これを利用してカウントを同期させている。

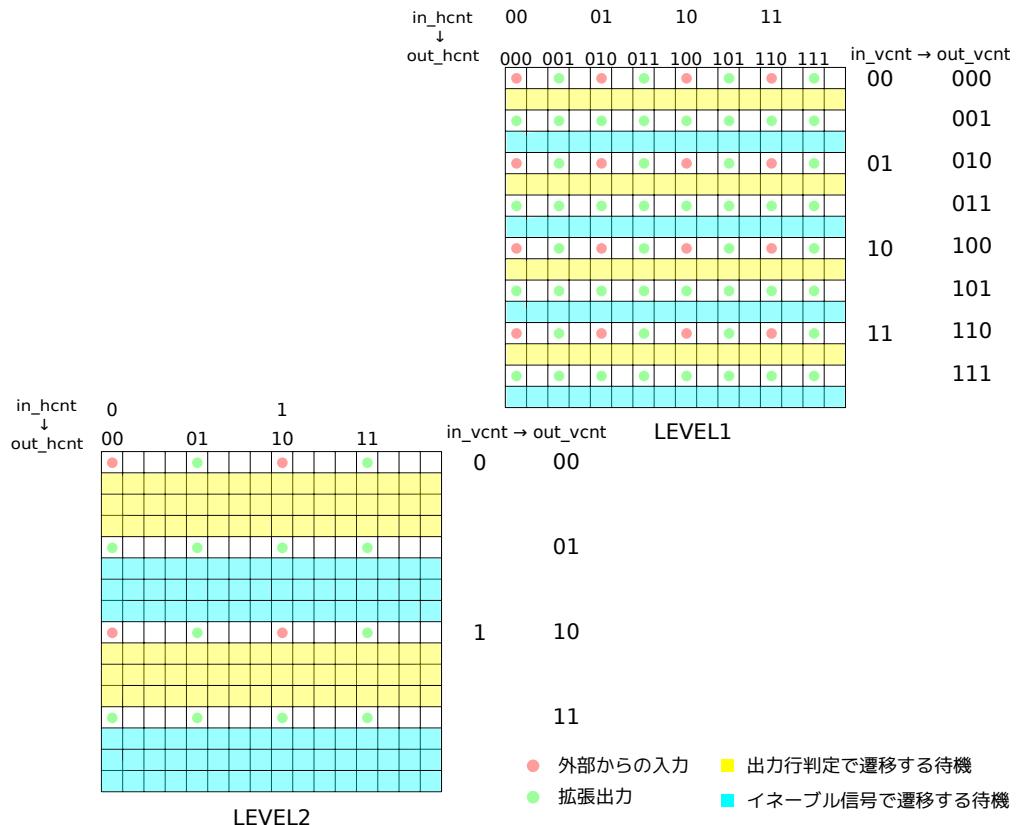


図 4.14: LEVEL ごとの状態遷移イメージ

また、pooling モジュールと同様に、座標値 (h_cnt, v_cnt) の出力はサイズの変化に合わせて適切に変更される必要がある。4.1.4 節で述べた通り、各 unpooling モジュールのサイズは入力に対してそれぞれ 2 倍となるので、適切な値による下位 1 ビットの拡張を行えばよい。図 4.14、表 4.1 に拡張方向に基づく座標値の変更方法を示す。

表 4.1: 拡張方向に基づく座標値の変更

	左 : h_cnt は元画素と同じ	右 : h_cnt は元画素+1
上 : v_cnt は元画素と同じ	h_cnt 0 ビット拡張 v_cnt 0 ビット拡張	h_cnt 1 ビット拡張 v_cnt 0 ビット拡張
下 : v_cnt は元画素+1	h_cnt 0 ビット拡張 v_cnt 1 ビット拡張	h_cnt 1 ビット拡張 v_cnt 1 ビット拡張

4.2.6 buf モジュール

buf モジュールは、ItgNet モジュールが受け取る unpooling からの出力とプーリング層適用前の特徴マップを対応付ける機能を持つ。通常ならば、それぞれの特徴マップ出力に必要なレイテンシの差分を求め、差分だけバッファリングさせることで出力のタイミングを合わせるのが一般的である。しかし、本システムはフィルタ演算を始めとする処理の差異が多く、レイテンシの差分を正確に求めるのが困難である。そこで、画像 1 枚分のメモリを確保し、座標値 (h_cnt, v_cnt) に基づくアドレスによって対応付けることとした。図 4.15 に buf モジュールの設計を示す。

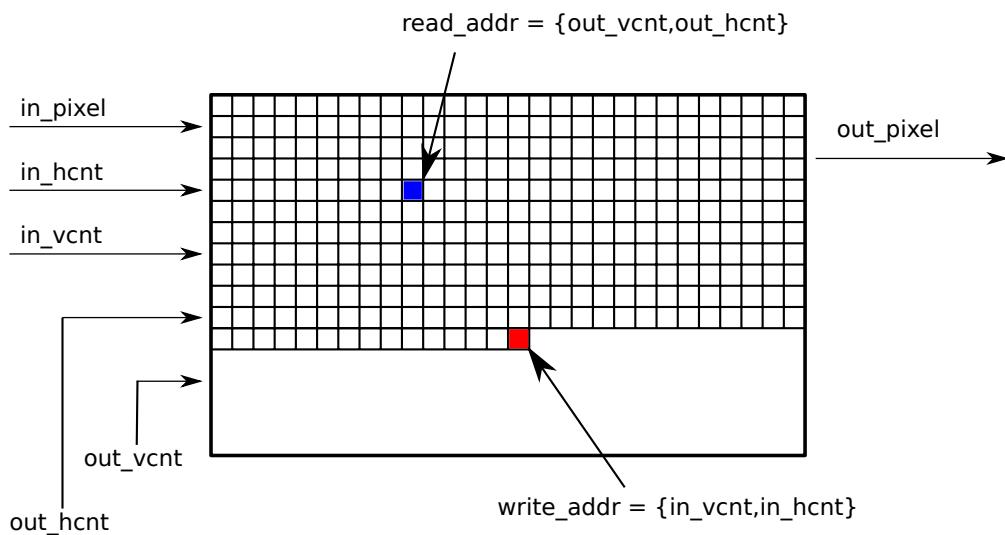


図 4.15: buf モジュール

out_hcnt , out_vcnt は各 ItgNet モジュール前段の unpoling モジュールの出力が渡される。これにより、unpooling モジュールからの画素値と同じ座標値を持つ pooling 前の画素値を対応付けることができた。

4.3 学習

4.3.1 ツール

学習には、Python 上で動作するニューラルネットワーク向けフレームワークである Chainer[11] を用いる。Chainer では、複雑なデータ構造を簡潔な記述で構築することができ、CUDA による GPU を用いた高速な学習も可能である。

また、学習及び評価における各処理には、数値計算ライブラリの Numpy や、画像処理ライブラリの を用いる。

4.3.2 学習データ

今回ニューラルネットワークの学習に用いる画像データと教師ラベルからなるデータセットは??枚であり、このうち??枚を訓練用データセット、??枚を評価用データセットとして学習を行う。画像データは長崎大学病院から提供された実際の手術画像であり、画像サイズは 672×528 となっている。また、それらの画像を医師が目視で判断し、手動でラベリングを行い作成された画像を教師データとした。これらの画像例を以下の図 4.16、図 4.17 に示す。



図 4.16: 実際の手術画像データ

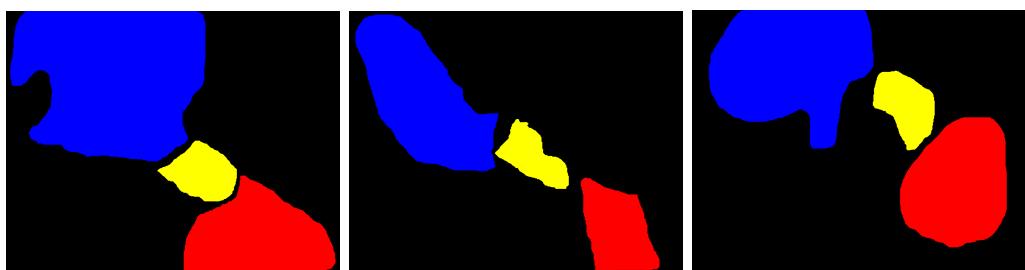


図 4.17: 教師データ

4.3.3 量子化手法

ソフトウェアでの実装においては、各演算は 32 ビットの浮動小数点型で行われ、学習の結果得られる重みやバイアス等も同じ型となっている。しかしながら、ハードウェア実

装においてパラメータに浮動小数点型を用いるのは資源容量的に厳しく、固定小数点型を利用するのが望ましい。

そのため、本来ならば学習においても固定小数点型を利用して学習を進め、パラメータを決定付けるべきだが、Chainer では浮動小数点型を用いることが前提となっており、固定小数点型で動作させるのは大変難しい。そこで、学習は浮動小数点型で行い、得られたパラメータを固定小数点型に変換して実装することとした。

第5章

評価と考察

5.1 評価

5.2 考察

図の参照、図 6.1

表の参照、表 6.1



図 5.1: Sample 図

表 5.1: Sample 表

	XXX	XX	XXXXX	XXXX	XXXX
XXXX	xxx	xxx	xxxxx	xx	xx
XXXXX	xxx	xxx	xxxxx	xx	xx

第6章

結論

図の参照、図 6.1

表の参照、表 6.1



図 6.1: Sample 図

表 6.1: Sample 表

	XXX	XX	XXXXX	XXXX	XXXX
XXXX	xxx	xxx	xxxx	xx	xx
XXXXX	xxx	xxx	xxxx	xx	xx

謝 辞

Thank you

参考文献

- [1] 公益社団法人全日本病院協会. 胆囊切除術患者に対する腹腔鏡下手術施行率, <https://www.ajha.or.jp/hms/qualityhealthcare/indicator/21/>.
- [2] 医療法人光生会. 腹腔鏡手術について, http://www.koseikai-hp.or.jp/kouseikai_index/kouseikai_clinicindex/01_koseikai_h_26/01_koseikai_h_261.html.
- [3] Intuitive — da vinci robotic assisted surgical systems : <https://www.intuitive.com>.
- [4] Taito Manabe, Koki Tomonaga, and Yuichiro Shibata. CNN Architecture for Surgical Image Segmentation Systems with Recursive Network Structure to Mitigate Overfitting. 2019.
- [5] 一般社団法人日本医療情報学会. 医療画像データ収集事業に用いる情報システム構築ガイドライン, http://jami.jp/about/documents/amed_report.pdf.
- [6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net:Convolutional Networks for Biomedical Image Segmentation. In Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention, pp.234-241. 2015.
- [7] AI を活用したリアルタイム内視鏡診断サポートシステム開発 : https://jpn.nec.com/press/201707/20170710_01.html.
- [8] 佐野 健太郎, 河野郁也, 中里直人, Alexander Vazhenin, Stanislav Sedukhin. FPGAによる津波シミュレーションの専用ストリーム計算ハードウェアと性能評価. 2015.
- [9] PASCAL VOC2011 Example Segmentations : <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/segexamples/>.
- [10] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. 2013.
- [11] Preferred networks, inc. chainer : A exible framework of neural networks : <http://chainer.org/>.