博士論文

進化計算法を用いたシングルフレーム超解像処理の自動構築

Automatic Construction of Single Frame Super-Resolution Using Evolutionary Computation

国立大学法人　横浜国立大学
大学院環境情報学府

夏井 裕介
Yusuke NATSUI

2018 年 3 月

あらまし

超解像技術とは，1 枚または複数枚の低解像度画像から高画質な高解像度画像を生成する技術である．近年，4K や 8K ディスプレイなどの高解像度ディスプレイの登場によって映像機器の高解像度化が進んでいるが，現在普及している動画像の多くはこの解像度を満たしていない．そのため，低解像度の動画像を表示する際には高解像度に変換する必要がある．しかし，単純に画像を拡大するだけでは高画質な動画像を得ることは難しい．そこで超解像技術が必要とされている．

超解像は，複数枚の低解像度画像を入力として処理を行うマルチフレーム超解像と 1 枚の低解像度画像を入力として処理を行うシングルフレーム超解像に大別される．本研究では，シングルフレーム超解像処理を対象とする．シングルフレーム超解像はマルチフレーム超解像と比較して，計算コストが低く，低解像度画像が 1 枚の場合にも適用することが可能なため，応用範囲が広く様々な研究がなされている．しかし，一般的にシングルフレーム超解像は画質と計算コストがトレードオフの関係となっており，高速に高精度な超解像処理を行うことは困難である．

この問題に対して，本研究では，局所的な画素情報から高解像度画像の画素値を比較的単純な関数を用いて算出する計算コストの低い処理方式を採用し，グラフ構造式を与えられた学習画像に合わせて進化計算法の 1 つである Cartesian Genetic Programming（CGP）を用いて設計することで，小規模な回路で，高速かつ高精度な超解像処理を自動構築する手法を目指す．また，本方式は画像の局所的な入力ごとに独立した処理を行うことが可能なため，GPU を用いた並列計算による高速化を行う．

まず，提案する手法のベースとなる CGP を用いたシングルフレーム超解像処理の構築手法を提案する．さらに，精度の向上を目的として，複数のグラフ構造式をアンサンブルする手法（以下，手法 1 と呼ぶ）を提案する．３カテゴリの傾向の異なる画像セットに対して超解像処理を適用し，従来の手法と比較して高速で高画質な超解像処理を構築できることを示す．

しかし，この手法は画素ごとに独立して処理を行っているため，拡大画像において周囲の画素との関係が考慮されていない．そこで次に，拡大された画像の周囲の画素の関係性を考慮する処理を用いた超解像処理の構築手法（以下，手法２と呼ぶ）を提案する．この提案手法では，拡大処理の後に，繰り返し補正を行う処理を行うことで，周囲の画素の関係性を考慮することができる．局所的な情報だけでは高精度に処理を行うことが難しい部分の精度向上を目指す．同様の３カテゴリの傾向の異なる画像セットに対して超解像処理実験を行い，繰り返し補正処理を適用し周囲の画素の関係性を考慮することの有効性を示す．

また，画像中には平坦な部分やエッジの部分など多様な領域が存在しているが，手法１，手法２は，すべての領域に対して同一の超解像処理を行っており，明示的に画像の領域に適した処理を行なっていない．そこで，画像の局所的な領域に応じた処理を適用する超解像処理の構築手法（以下，手法３と呼ぶ）を提案する．この提案手法では，低解像度画像に対して変換処理を行い，対応する拡大処理で処理を行う領域を抽出することでその領域に特化した超解像処理を行う．様々な領域に対して特化した処理を構築できていることを示すために，様々な被写体を含んだ自然画像セットに対して超解像処理実験を行い，その有効性を示す．

これまでに提案した手法は，それぞれ利点と欠点があり組み合わせることで精度の向上が期待で

きる．最後に，これまでに提案した手法１と手法２，手法２と手法３をそれぞれ組み合わせた２つの手法を提案する．そして近年，画像処理・認識など様々な分野でその有効性が示されている Deep Neural Networks（DNN）を用いた超解像処理手法と計算コストと画質を比較を行う．そして，提案手法を組み合わせることの有効性を示し，DNN を用いた手法との関係について述べる．

Abstract

Super-Resolution (SR) is a technique to generate a high quality high-resolution (HR) image from single and multiple low-resolution (LR) images. Recently, with the advent of HR displays such as 4K and 8K displays, the resolution of video equipment has been increasing, however, most of the images that are currently popular do not match the resolution of these displays. Therefore, when displaying LR images, it is necessary to convert it to HR images. However, it is difficult to obtain high-quality images by enlarging the images simply. Therefore, SR technology is required.

SR methods can be classified as multi-frame SR and single frame SR. In this research, we target single frame SR. Single frame SR is a technique to generate a HR image from one LR image. it needs only one input image and has lower computational cost than multi-frame SR. Additionally, it is more practical for a lot of applications, and is studied widely. In general, however, single frame SR has trade-off between image quality and computational cost.

To solve this problem, in this research, we adopt a method with low computation cost to calculate the pixel values of a HR image from the local pixel information using simple functions. We propose a method for automatic construction of high speed and high precision SR operation that can be implemented with a small circuit, by realizing SR using Cartesian Genetic Programming (CGP) in accordance with training images. In addition, since this method processes each image patch of the LR image independently, it can speed up by parallel computation using the GPU.

First, we propose a construction method of single frame SR operation using CGP which is the base of our proposed methods. Furthermore, to improve image quality, we employ an ensemble method of multiple graph structured programs (Method 1). We test our method for three different categorical image sets. Experiment shows that our method constructs SR operation with better trade-off between speed and image quality than the conventional methods.

However, since this method independently estimates each pixel of the enlarged image, the relation with neighboring pixels of it is not considered. Next, we propose a SR construction method (Method 2) using recursive operation to consider the relationship between neighboring pixels of the enlarged image. It aim to improve the accuracy of the image segment where it is difficult to generate HR pixels precisely with only local information. In experiment for the image sets of three different categories, we show the effectiveness of this method considering the relationship of neighboring pixels by applying recursive operation.

Although there are various regions such as a flat part and an edge part in the image, Method 1 and Method 2 perform the same SR operation for all local regions of LR images. These methods do not explicitly perform suitable operation for the image regions. Therefore, we propose a method that can learn several image conversions for specific upscaling that are optimal for different patches (Method3). Method3 is a single frame SR method using multiple graph structured programs based on CGP. In Method3, a conversion graph is performed on a LR image and extracts a region to be processed by the corresponding upscaling graph, thereby the constructed SR operation is specialized for each region of LR image. In

order to show that specialized processing can be constructed for various areas, we test our method to a natural image set containing various subjects and show effectiveness of our method.

Every method proposed so far has advantages and disadvantages, respectively, and can be expected to improve accuracy by combining them. Finally, we propose two methods combining Methods 1 and 2, and Methods 2 and 3 respectively. In recent years, Deep Neural Networks (DNN) has shown the effectiveness in various fields such as image processing and recognition. Therefore, we compare our methods with the SR method using DNN in evaluation of computational cost and image quality. We show the effectiveness of combining our proposed methods and the difference of characteristics from the method using DNN.

目次

第 1 章 序論 1
- 1.1 背景と目的 . . . 1
- 1.2 本論文の構成 . . . 2

第 2 章 関連研究 3
- 2.1 進化計算法 . . . 3
  - 2.1.1 概要 . . . 3
  - 2.1.2 遺伝的アルゴリズム（Genetic Algorithm; GA） . . . 3
  - 2.1.3 遺伝的プログラミング（Genetic Programming; GP） . . . 5
  - 2.1.4 Cartesian Genetic Programming（CGP） . . . 5
- 2.2 超解像処理 . . . 7
  - 2.2.1 概要 . . . 7
  - 2.2.2 シングルフレーム超解像の従来研究 . . . 7
  - 2.2.3 従来研究の課題と進化計算法を用いた超解像処理 . . . 9
- 2.3 まとめ . . . 9

第 3 章 複数のグラフ構造式を組み合わせた超解像処理の自動構築 11
- 3.1 はじめに . . . 11
- 3.2 CGP を用いた拡大処理 . . . 11
  - 3.2.1 概要と特徴 . . . 11
  - 3.2.2 処理の流れ . . . 11
- 3.3 複数のグラフ構造式を組み合わせた超解像処理 . . . 12
  - 3.3.1 概要と特徴 . . . 12
  - 3.3.2 補正処理 . . . 12
  - 3.3.3 全体の超解像処理の流れ . . . 13
  - 3.3.4 グラフ構造式の最適化 . . . 13
  - 3.3.5 GPU を用いた並列処理による高速化 . . . 14
- 3.4 超解像処理実験 . . . 14
  - 3.4.1 実験設定 . . . 14
  - 3.4.2 実験結果と考察 . . . 18
- 3.5 まとめ . . . 25

第 4 章 周囲の画素の関係性を考慮したシングルフレーム超解像処理の自動構築 26
- 4.1 はじめに . . . 26
- 4.2 周囲の画素の関係性を考慮した超解像処理 . . . 26
  - 4.2.1 概要と特徴 . . . 26
  - 4.2.2 補正処理 . . . 27

4.2.3 処理の流れ . . . 27
4.3 超解像処理の最適化 . . . 28
4.3.1 GPU を用いた並列処理による高速化 . . . 28
4.4 超解像処理実験 . . . 28
4.4.1 実験設定 . . . 28
4.4.2 実験結果と考察 . . . 30
4.4.3 第 3 章の手法との比較 . . . 36
4.5 まとめ . . . 38

第 5 章 画像の領域に適した超解像処理の自動構築 39
5.1 はじめに . . . 39
5.2 画像の領域に適した超解像処理 . . . 39
5.2.1 概要と特徴 . . . 39
5.2.2 処理の流れ . . . 40
5.2.3 超解像処理の最適化 . . . 40
5.3 GPU を用いた並列処理による高速化 . . . 42
5.4 超解像処理実験 . . . 42
5.4.1 実験設定 . . . 42
5.4.2 実験結果と考察 . . . 45
5.4.3 第 4 章の手法との比較 . . . 50
5.5 まとめ . . . 52

第 6 章 提案手法の組み合わせと Deep Learning を用いた手法との比較 53
6.1 はじめに . . . 53
6.2 提案手法の組み合わせと Deep Learning を用いた手法との比較 . . . 53
6.2.1 概要と特徴 . . . 53
6.2.2 学習画像のサンプリング . . . 55
6.2.3 超解像処理の最適化 . . . 55
6.3 超解像処理実験 . . . 55
6.3.1 実験設定 . . . 55
6.3.2 実験結果と考察 . . . 58
6.4 まとめ . . . 62

第 7 章 結論 63
7.1 本論文で得られた成果 . . . 63
7.2 今後の課題 . . . 64

謝辞 65

参考文献 66

本研究に関する発表 70

付録 A 獲得されたグラフ構造 72

図目次

2.1 GA の処理の流れ 4
2.2 交叉の例 5
2.3 突然変異の例 5
2.4 GP の木構造の例 5
2.5 CGP の表現型と遺伝子型の対応 6
2.6 Chang らの手法の例 10
2.7 Sparse Cording の例 10

3.1 S=2, I=3 のときの拡大処理の概要 12
3.2 拡大処理と補正処理の組み合わせた処理の概要 13
3.3 補正処理の学習の概要 14
3.4 テスト画像の超解像処理結果（Face） 21
3.5 テスト画像の超解像処理結果（Building） 21
3.6 テスト画像の超解像処理結果（Text） 22
3.7 構築された拡大処理（Face） 23
3.8 構築された 1 つ目の補正処理（Face） 23
3.9 構築された 2 つ目の補正処理（Face） 24

4.1 提案手法の超解像処理の概要 27
4.2 テスト画像の超解像処理結果（Face） 34
4.3 テスト画像の超解像処理結果（Building） 34
4.4 テスト画像の超解像処理結果（Text） 35
4.5 テスト画像の超解像処理結果（Face） 37
4.6 テスト画像の超解像処理結果（Building） 37
4.7 テスト画像の超解像処理結果（Text） 37

5.1 画像中の局所的なパッチの例 39
5.2 提案手法の概要 40
5.3 テスト画像結果の一部（Zebra） 46
5.4 テスト画像結果の一部（Wall man） 46
5.5 テスト画像結果の一部（Woman） 47
5.6 テスト画像に対する各処理の出力結果 48
5.7 各処理の 5×5 カーネルの入力位置 49
5.8 テスト画像の超解像処理結果（Man） 51
5.9 テスト画像の超解像処理結果（Building） 51

6.1 手法 1 と手法 2 の組み合わせ 54

6.2 手法３と手法２の組み合わせ 54
6.3 テスト画像結果の一部（Baby from Set5） 59
6.4 テスト画像結果の一部（Lenna from Set14） 59
6.5 テスト画像結果の一部（Zebra from Set14） 60
6.6 テスト画像結果の一部（Baboon from Set14） 60
6.7 画質（Set5）と積和演算数の関係 61

A.1 3 章の実験において獲得された CGP_{enl} の構造（Building） 72
A.2 3 章の実験において獲得された $CGP_{up}1$ の構造（Building） 73
A.3 3 章の実験において獲得された $CGP_{up}2$ の構造（Building） 74
A.4 3 章の実験において獲得された CGP_{enl} の構造（Text） 74
A.5 3 章の実験において獲得された $CGP_{up}1$ の構造（Text） 75
A.6 3 章の実験において獲得された $CGP_{up}2$ の構造（Text） 75
A.7 4 章の実験において獲得された $CGP_{mod}1$ の構造（Face） 76
A.8 4 章の実験において獲得された $CGP_{mod}1$ の構造（Building） 76
A.9 4 章の実験において獲得された $CGP_{mod}1$ の構造（Text） 77
A.10 5 章の実験において獲得された CGP_{enl} の構造 78
A.11 5 章の実験において獲得された CGP_{up1} の構造 79
A.12 5 章の実験において獲得された CGP_{conv1} の構造 80
A.13 5 章の実験において獲得された CGP_{up2} の構造 81
A.14 5 章の実験において獲得された CGP_{conv2} の構造 82
A.15 5 章の実験において獲得された CGP_{up3} の構造 83
A.16 5 章の実験において獲得された CGP_{conv3} の構造 84
A.17 5 章の実験において獲得された $CGP_{up}4$ の構造 85
A.18 5 章の実験において獲得された $CGP_{conv}4$ の構造 85
A.19 5 章の実験において獲得された $CGP_{up}5$ の構造 86
A.20 5 章の実験において獲得された $CGP_{conv}5$ の構造 86

表目次

3.1 CGP の設定 16
3.2 CGP で用いた演算子 17
3.3 比較手法の設定 17
3.4 CRFCN の GA のパラメータ設定 17
3.5 各手法の SSIM の評価 19
3.6 各手法の PSNR の評価 19
3.7 テスト画像の処理時間の比較 19
3.8 学習とテストで画像の傾向を変えた場合の PSNR，SSIM 19
3.9 Best, Average（Avg.）and Standard deviation (Std.) of SSIM 20
3.10 Best, Average（Avg.）and Standard deviation (Std.) of PSNR 20

4.1 CGP の設定 29
4.2 CGP で用いた演算子 30
4.3 比較手法の設定 31
4.4 各手法の SSIM の評価 31
4.5 各手法の PSNR の評価 32
4.6 各手法の画像ごとの SSIM の標準偏差と平均順位 32
4.7 各手法の画像ごとの PSNR の標準偏差と平均順位 33
4.8 3 つの画像セット（Face，Building，Text）に対するオープン評価 33
4.9 CPU と GPU における超解像処理の処理時間 33
4.10 各手法の PSNR の評価 36
4.11 各手法の SSIM の評価 36

5.1 CGP の設定 44
5.2 CGP で用いた演算子 44
5.3 比較手法の設定 44
5.4 各手法の PSNR の評価 45
5.5 各手法の SSIM の評価 45
5.6 テスト画像の処理時間の比較（sec） 45
5.7 各手法の PSNR の評価 50
5.8 各手法の SSIM の評価 50

6.1 CGP の設定 57
6.2 CGP で用いた演算子 57
6.3 比較手法の設定 57
6.4 各手法の PSNR の評価値と積和演算数 58

第1章　序論

1.1　背景と目的

超解像技術とは，1 枚または複数枚の低解像度画像から高画質な高解像度画像を生成する技術である．近年，4K や 8K ディスプレイなどの高解像度ディスプレイの登場によって映像機器の高解像度化が進んでいるが，現在普及している動画像の多くはこの解像度を満たしていない．そのため，低解像度の動画像を表示する際には高解像度に変換する必要がある．しかし，単純に画像を拡大するだけでは高画質な動画像を得ることは難しい．そこで超解像技術が必要とされている．

超解像は，複数枚の低解像度画像を入力として処理を行うマルチフレーム超解像と 1 枚の低解像度画像を入力として処理を行うシングルフレーム超解像に大別される．シングルフレーム超解像はマルチフレーム超解像と比較して，計算コストが低く，低解像度画像が 1 枚の場合にも適用することが可能なため，応用範囲が広く様々な研究がなされている．本研究では，このシングルフレーム超解像処理を対象とする．しかし，一般的にシングルフレーム超解像は画質と計算コストがトレードオフの関係となっており，高速に高精度な超解像処理を行うことは困難である．

この問題に対して，本研究では，局所的な画素情報から高解像度画像の画素値を比較的単純な関数を用いて算出する計算コストの低い処理方式を採用し，グラフ構造式を与えられた学習画像に合わせて進化計算法の 1 つである Cartesian Genetic Programming（CGP）を用いて設計することで，小規模な回路で，高速かつ高精度な超解像処理を自動構築する手法を目指す．また，本方式は画像の局所的な入力から独立した処理を行うため並列計算が可能であるため，GPU を用いた高速化を行う．

1.2 本論文の構成

本論文は 7 章から構成されている．本論文の構成は次の通りである．まず，第 2 章で本研究に関する従来研究について述べる．第 3 章では，CGP を用いたシングルフレーム超解像処理の構築手法と複数のグラフ構造式をアンサンブルする手法を提案し，その有効性を検証する．第 4 章では，周囲の画素の関係性を考慮することができる超解像処理の構築手法を提案し，その有効性を検証する．第 5 章では，画像の局所的な領域に応じた処理を適用する超解像処理の構築手法を提案し，その有効性を検証する．第 6 章では，これまでの提案手法を組み合わせた手法と Deep Learnning を用いた超解像処理手法との比較を行い，その関係について述べる．最後に第 7 章で，本論文のまとめと今後の課題について述べる．

第2章　関連研究

本章では，本研究に関連が深い進化計算法，超解像処理の概要，シングルフレーム超解像の従来研究とその課題について述べる．

2.1　進化計算法

2.1.1　概要

進化計算法（Evolutionary Computation; EC）とは，生物の進化に着想を得た計算手法のことであり，組み合わせ最適化問題の解の探索などに用いられている．進化計算法の代表的な手法として遺伝的アルゴリズム（Genetic Algorithm; GA）[1, 2]，遺伝的プログラミング（Genetic Programming; GP）[3]などが挙げられる．これらは生殖，突然変異，自然淘汰などの生物の進化の過程を模倣し，与えられた問題に対して最適解を求める手法である．本節では，本研究と関連の深い GA と GP，本研究で用いている Cartesian Genetic Programming（CGP）[4] について述べる．

2.1.2　遺伝的アルゴリズム（Genetic Algorithm; GA）

GA は，John. H. Holland によって提案された最適化，探索を行うアルゴリズムである．GA では，個体を文字列の染色体で表しその染色体の構造を遺伝子型（Genotype），遺伝子型を問題の解の形に変換したものを表現型（Phenotype）という．探索を行う際には複数の個体からなる個体集団を用意し遺伝操作によって新たな個体を生成し，世代交代を繰り返し行うことで良い個体を残し，最適解を探索する手法である．遺伝操作には選択（Selection），交叉（Crossover），突然変異（Mutation）がある．個体の評価には，ある個体が対象とする問題に対してどの程度優れているかを表す適応度を用いる．GA の処理手順を図 2.1 に示す．

選択（Selection）

選択は世代交代において適応度に応じて個体を選ぶ操作である．選択操作によって適応度の高い個体が個体が生き残り，低い個体は淘汰される．選択方法には，個体の適応度に比例した確率で個体を選択するルーレット選択や決められた数の個体を個体集団の中からランダムに選択し，その中の最も適応度の高い個体を選択するトーナメント選択，個体集団の中の個体を適応度が高い順に順位付けを行いそれぞれの順位にあらかじめ決めておいた確率を用いて個体を選択するランク選択などがある．またこれらの選択方法と組み合わせて適応度の最も高い個体を選択するエリート選択を用いる場合がある．

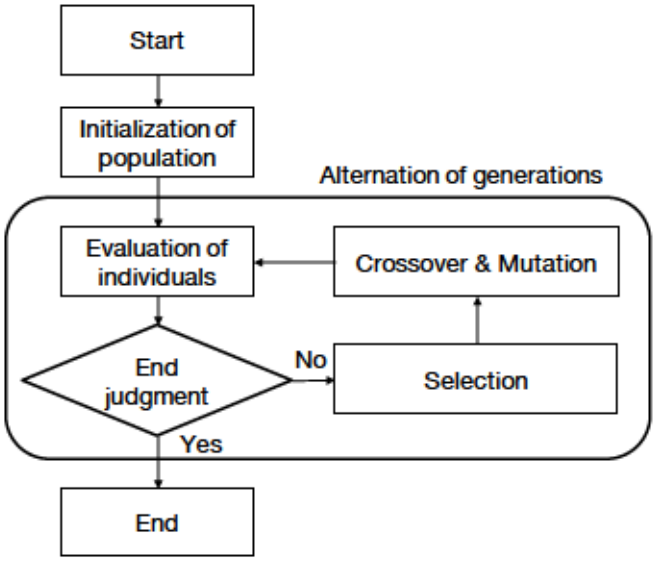

図 2.1: GA の処理の流れ

交叉 （Crossover）

交叉とは，２つの個体に対して一定の確率で一部の遺伝子を交換し，子個体を生成する操作である．交叉を行うことで２つの親個体からより優れた子個体が生成されることを期待している．交叉の方法には，１点交叉，２点交叉，一様交叉などがある．１点交叉，一様交叉の例を図 2.2 に示す．１点交叉は，ランダムに交叉点を１つ選びその点から後ろの遺伝子を個体間で交換する交叉方法である．２点点交叉は，ランダムに交叉点を２つ選び交叉点ではさまれた部分を個体間で交換する交叉方法である．また一様交叉は，各遺伝子を確率的に交換を行う交叉方法である．

突然変異 （Mutation）

突然変異とは，一定の確率で一部の遺伝子を変化させる操作である．突然変異を行うことで，交叉だけでは生成することができない個体を生成することができ探索範囲を広げることができる．また局所解から抜け出す効果がある．突然変異の例を図 2.3 に示す．

世代交代（Alternation of generations）

選択，交叉，突然変異の遺伝操作によって新たな個体を生成し個体集団を更新することを世代交代という．世代交代の方法には，個体集団の多様性の維持に有効とされている Minimal Gneration Gap（MGG）[5]が提案されている．MGG は個体集団から異なる２個体を親個体として選択し，親個体を交叉，突然変異することによって子個体を決められた数だけ生成する．子個体の中から適応度の最も高い個体とルーレット選択によって選ばれた個体をそれぞれ親個体と入れ替えることで世代交代を行う．

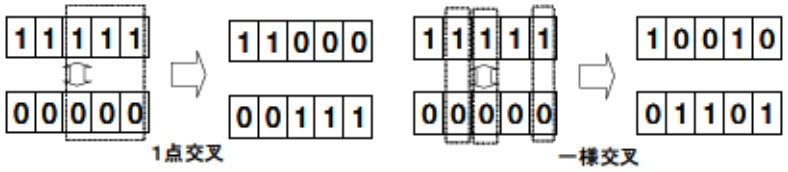

図 2.2: 交叉の例

図 2.3: 突然変異の例

2.1.3 遺伝的プログラミング（Genetic Programming; GP）

GP は，John R. Koza によって提案された GA を拡張した木構造を個体の表現形式とする構造最適化手法である．GP の個体の木構造の例を図 2.4 に示す．GA と同様に，交叉や突然変異などの遺伝操作や世代交代を繰り返すことによって与えられた問題に適した木構造を生成する．木構造によって表現される個体には，関数やプログラムなどがある．GP の交叉方法には，2 つの個体からランダムに選択された部分木の交換がある．突然変異ではあるノードや部分木を新しいノードや部分木に交換する．GP の GA との大きな違いは，遺伝操作を行うとノードの数が変化することが挙げられる．そのため，世代交代を繰り返していくうちに構造が巨大化するブロートが起こる可能性がある．このブロートを抑制するために，従来研究では構造の大きさに制限を設けている．

2.1.4 Cartesian Genetic Programming（CGP）

CGP は Julian F. Miller によって提案されたグラフ構造を表現型とする構造最適化手法である．CGP の表現型と遺伝子型の対応を図 2.5 に示す．CGP の構造はフィードフォワード型のグラフ構造で表現され，入力ノード，演算ノード，出力ノードから構成される．入力ノードは数式への入力値に対応し，出力ノードから数式の出力値を得る．CGP では表現型のグラフ構造をノードの種

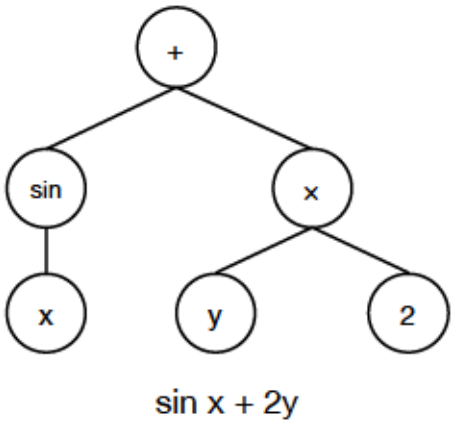

図 2.4: GP の木構造の例

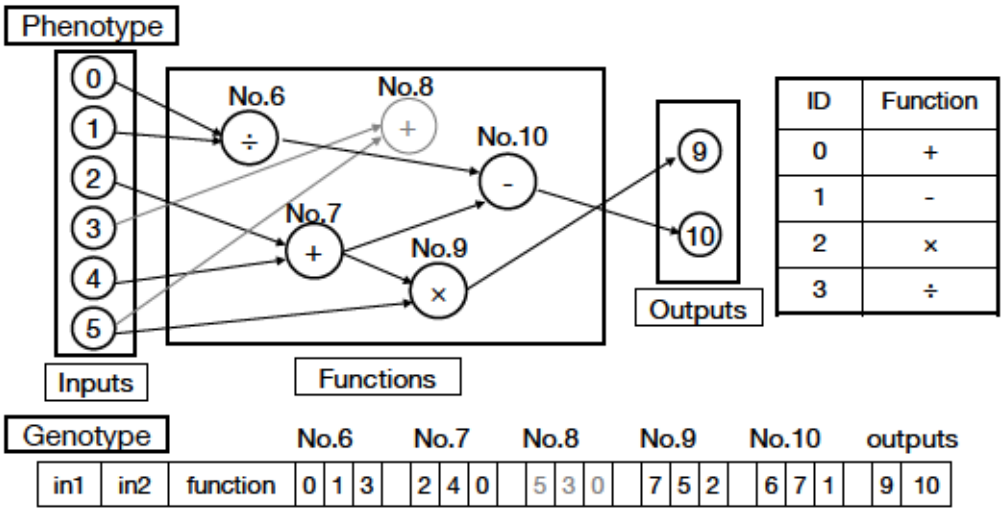

図 2.5: CGP の表現型と遺伝子型の対応

類，接続元を記述した一次元の文字列の遺伝子型に変換し，交叉や突然変異などの遺伝操作を行うことでグラフ構造を構築する．入力ノード，演算ノード，出力ノードの順に番号が割り当てられており，接続元として自分のノード番号より小さい番号を選択することで，フィードフォワード構造を実現している．フィードフォワード型のグラフ構造を採用する利点としては，ノードの再利用が可能となることが挙げられる．CGP の全ノード数は固定されており，染色体の遺伝子長は固定長になる．しかし遺伝子型から表現型に変換する際，出力ノードに最終的につながらないノードは表現型に表れないため，グラフ構造のノード数は表現上は可変となる．

CGP は GP と同様に関数やプログラムを表現することができ，画像処理や分類処理の構築に応用する研究が多くなされている [6, 7, 8, 9]．画像処理の構築の例では，Harding らは，画像処理ソフト GIMP [10]で用いられている 10 種類の画像処理フィルタを CGP を用いて構築し，画像処理フィルタを再現することが可能であることを示した [8]．

2.2 超解像処理

2.2.1 概要

超解像には大きく分けて，複数枚の低解像度画像を入力として処理を行うマルチフレーム超解像 [11, 12]と 1 枚の低解像度画像を入力として処理を行うシングルフレーム超解像 [13]がある．マルチフレーム超解像は，複数枚の同じ被写体が写った低解像度画像から標本点を集めることで高解像度画像を生成する手法であり，条件が合えば高解像度画像の真値を復元することができる．しかし，1 ピクセル未満のサブピクセル単位での位置ずれがある複数枚の低解像度画像が必要となり，処理する際にも高精度な位置合わせが必要となるため処理に時間がかかってしまう．一方シングルフレーム超解像は，真値を復元することができる保証はないがマルチフレーム超解像と比べて計算コストが低く，1 枚の画像から処理することができるため応用範囲が広く様々な手法が提案されている．本研究でも，シングルフレーム超解像を対象としている．シングルフレーム超解像は，大きく Interpolation-based，Reconstruction-based，Learning-based の 3 つのアプローチに分けることができる．

2.2.2 シングルフレーム超解像の従来研究

Interpolation-based

Interpolation-based アプローチ [13, 14, 15, 16]は，画像を拡大した際に足りない画素を周囲の画素から補間することで超解像処理を行う手法である．Bilinear や Bicubic，Lanczos [17, 13]などの関数を用いて周囲の画素の距離に応じて重みを付け畳み込みを行う手法が挙げられる．1 次元の Lanczos 関数の例を式（2.1）に示す．

$$\mathrm{Lanczos}(x) = \begin{cases} \dfrac{r\sin(x\pi)\sin(x\pi/r)}{x^2\pi^2}, & (|x| < r) \\ 0, & (|x| \geq r) \end{cases} \tag{2.1}$$

ここで，x，r はそれぞれ周囲の画素までの距離，Lanczos 関数の半径を表している．Lacnzos 関数による補間は，リサンプリングを行う方式であるため，画像の高周波数成分を再現することができない．また，局所共分散の双対性を利用した New edge-directed interpolation（NEDI）[14]も提案されている．NEDI は，Bilinear，Bicubic などと比較すると補間された局所的なエッジをシャープに保つことができる．Interpolation-based アプローチは周囲の画素だけを用いて処理を行うため，計算コストは低いが，処理が単純なためエッジなどがぼけてしまい画質は低くなりやすい．

Reconstruction-based

Reconstruction-based アプローチ [18, 19, 20, 21]は，入力画像から仮の高解像度画像を推定し，それを縮小したものと入力画像との誤差を最小化するように高解像度画像を更新していく手法である．Sun らは自然画像とその縮小画像から抽出したエッジの勾配特性に関する事前知識を仮定して超解像処理を行う手法を提案している [19, 20]．この手法は低解像度画像のエッジ強度分布から事前知識を用いて高解像度画像のエッジ強度分布を推定し，出力画像のエッジ強度分布と推定した分布の誤差を最小化することでエッジ部分を良好に処理することができている．Interpolation-based

と同様に計算コストが低く，高速に処理することが可能であるが，事前知識に依存してしまい特定のパターン以外では高画質な画像を生成することが困難となっている．

Learning-based

Learning-based アプローチは，機械学習を用いて低解像度画像と高解像度画像のペアから学習し，その結果から釣果像処理を行う手法である．Learning-based アプローチは，さらに Example-based アプローチと NeuralNet-base のアプローチに分類される．Example-based アプローチは，低解像度画像と高解像度画像のパッチのペアを事例として保持しておき，その関係を用いて超解像処理を行う手法であり，数多くの研究がされている［22, 23, 24, 25, 26, 27］．Freeman らは低解像度と高解像度画像のパッチの関係を Markov network を用いて学習し，入力画像から高解像度画像を推定する手法を提案している［23］．高精度な処理を行うためには Freeman らの手法は大量の低解像と高解像パッチのペアが必要となり，探索に計算コストが掛かってしまう．Chang らは多様体学習のひとつである Locally Linear Embedding（LLE）を用いた手法を提案している［25］．Chang らの手法の概要を図 2.6 に示す．この手法は，拡大する低解像パッチをその近傍パッチの線形和で表現し，対応する高解像パッチを高解像度の近傍パッチの線形和から生成することで超解像処理を行っている．これによって事例として保持していないパッチに対しても処理を行うことができ，保持している事例の数を少なくすることができる．さらに効率よく少ない数の事例から超解像処理を行うことを目的として Sparse Cording（SC）を用いた手法が提案されている［26, 27］．SC は画像中のあるパッチをできるだけ少ない基底パッチ（Basis patch）の線形和で表す手法である．SC の例を図 2.7 に示す．これによって従来よりさらに保持している事例の数を少なくし，超解像処理を行うことが可能となっている．一般的に先に述べた 2 つのアプローチと比べて，Example-based アプローチは画質は良いが事例の探索などに処理時間がかかってしまう．

また，Glasner らはマルチフレーム超解像と Example-based アプローチを組み合わせた手法を提案している［28］．これは，従来の手法のようにあらかじめ事例を保持しておくのではなく，自己学習を行い，1 枚の入力画像とその縮小画像から生成された事例を用い，また画像中の共通領域を複数フレームとみなして超解像処理を行う手法である．高画質な画像を生成することができ，現状で最高レベルの手法として知られているが処理が非常に複雑であるため計算コストが大きい．Huang らは，射影幾何変換と局所的なアフィン変換を用いることで自己学習手法を改良した手法を提案している［29］. 従来の自己学習手法よりも多様なパッチを探索することができ，画質を向上することを示している．

NeuralNet-based アプローチは，低解像度画像や低解像度画像を bicubic などの補間手法で拡大した画像を入力として Convolutional Nerual Networks を用いて高解像度画像を生成する手法である．近年では，Deep Convolutional Neural Networks (ConvNets) を用いた超解像手法も提案されている［30, 31, 32, 33, 34, 35, 36, 37］．Dong らは，ConvNets を低解像度画像から高解像度画像の変換に適用し，Sparce Cording 手法のフレームワークと等価な処理を end-to-end で学習する SRCNN を提案している．［30, 31］SRCNN は，従来の手法より画質と処理速度において優れていることを示している．Kim らは，小さいフィルタと深いネットワーク構造を用いた Very Deep Super-Resolution（VDSR）［32］を提案している．VDSR は，ResNet［38］の手法を採用し SRCNN から Layer 数を 20 層に増やした ConvNets を使用した手法である．学習係数を SRCNN の 1 万倍に設定することで，SRCNN より少ない学習回数で収束し，高精度に超解像処理を行えることを示している．Dong らは，処理時間の削減を目的として SRCNN を改良した Fast-SRCNN（FSRCNN）［35］も提案している．FSRCNN は，SRCNN の Bicubic の前処理部分を無くし，中間層を増やしチャンネル数を調整する

ことや，従来より小さいフィルタサイズを用いることで処理時間の削減を行っている手法である．Kim らは，ネットワークの複雑さを減らし学習しやすくするために，重みを共有した Convolution を繰り返し行う Deeply-Recursive Convolutional Network を用いた手法（DRCN）[36]を提案している．Tai らは，グローバルとローカルの Residual 構造と重みを共有した Convolution を用いた Deep Recursive Residual Network（DRRN）[39]を提案している．重みを共有することで少ないパラメータ数で学習を行いやすくし 52 層の深さを実現している．Lai らは，Laplacian pyramid フレームワークのネットワークを用いた Laplacian Pyramid Super-Resolution Network（LapSRN）[37, 40]を提案している．LapSRN は，各 Pyramid レベルで重みを共有することでパラメータ数を削減している．

これらの NeuralNet-based アプローチは，Example-based アプローチと同様に，先に述べた 2 つの手法と比べて画質の良い画像を出力することができる．また，Example-based アプローチと比べて，事例を探索する必要がなく高速に超解像処理を行うことができるが，大量の積和演算を行うため，リアルタイム処理をするためには，GPU などの大量の積和演算を並列に実行することができる演算装置が必要となる．

2.2.3 従来研究の課題と進化計算法を用いた超解像処理

これまでに述べたようにシングルフレーム超解像処理は画質と計算コストがトレードオフの関係となっているという課題がある．筆者らの研究グループでは，この課題を解決するために進化計算法を用いた超解像処理手法を提案されている [41]．文献 [41]では，Bilinear によって拡大された低解像度画像をセルラ状に配置した進化型ニューラルネットワークによって高画質化処理を行う手法を提案している．この手法は，演算部分にシグモイド関数，線形関数，ステップ関数などの関数を用いており，画質と計算コストのトレードオフを考えると良好な結果を示している．

2.3 まとめ

本章ではまず進化計算法の手法として GA，GP および CGP について述べた．次に超解像処理の概要，シングルフレーム超解像の従来研究とその課題について述べた．次章からは本章で述べた進化計算法の 1 つである CGP を用いた小規模なグラフ構造式から構成される超解像処理の自動構築手法の提案を行う．

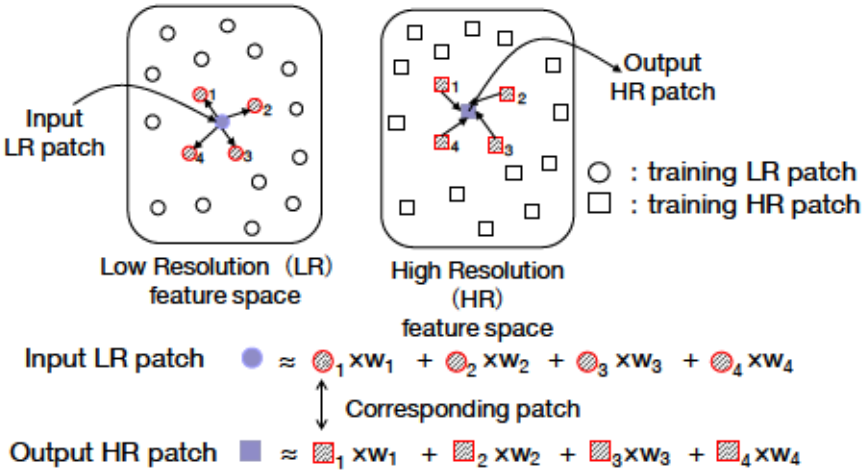

図 2.6: Chang らの手法の例

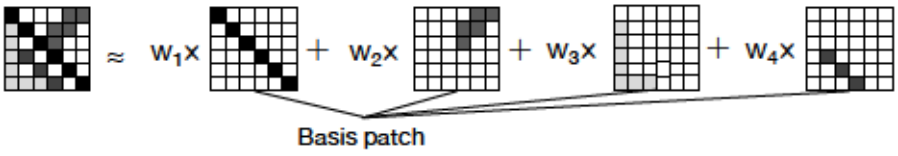

図 2.7: Sparse Cording の例

第3章　複数のグラフ構造式を組み合わせた超解像処理

3.1　はじめに

第 2 章で述べたようにシングルフレーム超解像処理は計算コストと画質がトレードオフの関係となっている．ConvNets を用いた超解像処理手法は他の手法と比較してこのトレードオフを解消することができているが，大量の積和演算を並列計算することができる GPU を用いて処理時間を削減している．そこで，本手法では小規模な回廊構造で超解像処理を構築することで，低計算コストかつ高画質なシングルフレーム超解像処理の構築を目指す．簡単な演算子の組み合わせを進化計算法の 1 つである CGP を用いてグラフ構造式（Graph Structured Program; GSP）を構築する．また，低解像度画像である入力画像の近傍画素のみを入力として高解像度画像の画素を推定する．そのとき，各近傍入力は独立して計算が可能であるため，並列計算による高速化も期待できる．本章では，基本となる 1 つのグラフ構造式を用いた拡大処理手法と，複数のグラフ構造式をアンサンブルした手法を提案し，超解像処理実験による性能評価を行う．

3.2　CGP を用いた拡大処理

3.2.1　概要と特徴

本手法は，低解像度画像を入力画像として，拡大された高解像度画像を出力する．入力画像の 1 つの画素およびその近傍の画素値と対応する高解像度画像の画素値との関係を CGP を用いて学習し，グラフ構造式を構築する．それを用いて画像を拡大する．

本手法の特徴として次の 3 点が挙げられる．

- 高解像度と低解像度画像のペアから CGP を用いて簡単な演算子の組み合わせで超解像処理を構築することで高精度な処理を行う．
- Example-based の手法のようにパッチの探索や最適化する必要がなく計算コストが低い
- 画素ごとに独立して計算できるため並列計算が可能

3.2.2　処理の流れ

画像を S 倍に拡大するために，低解像度画像の 1 つの画素を中心とする $I \times I$ [pixels] の近傍画素の画素値を入力として，$\mathrm{CGP}_{\mathrm{enl}}$ を用いて出力画像の S^2 個の画素の画素値を出力する．$S = 2$, $I = 3$ としたときのある 1 つの画素の $\mathrm{CGP}_{\mathrm{enl}}$ の例を図 3.1 に示す．3×3 [pixels] の画素ブロックを入力し，演算を行って 4 （2×2）つの出力値を算出する．この処理によって中心画素を 2 倍に拡大している．最終的な出力は，出力値が入力画像の画素値から大きく外れてしまうことを防ぐた

め，S^2 個の画素の出力値の平均値を入力画素値に近づけるように各出力値を調整する．調整処理は，CGP の i 番目の出力値を o_i，入力の中心画素値を C，最終的な i 番目の出力値を O_i として，次の式 (3.1) を用いて行われる．この調整処理によって，グラフ構造式の探索範囲を狭め，学習を簡単にすることが期待できる．

$$O_i = o_i + C - \frac{\sum_{j=1}^{S^2} o_j}{S^2} \ (i = 1, \ldots, S^2) \tag{3.1}$$

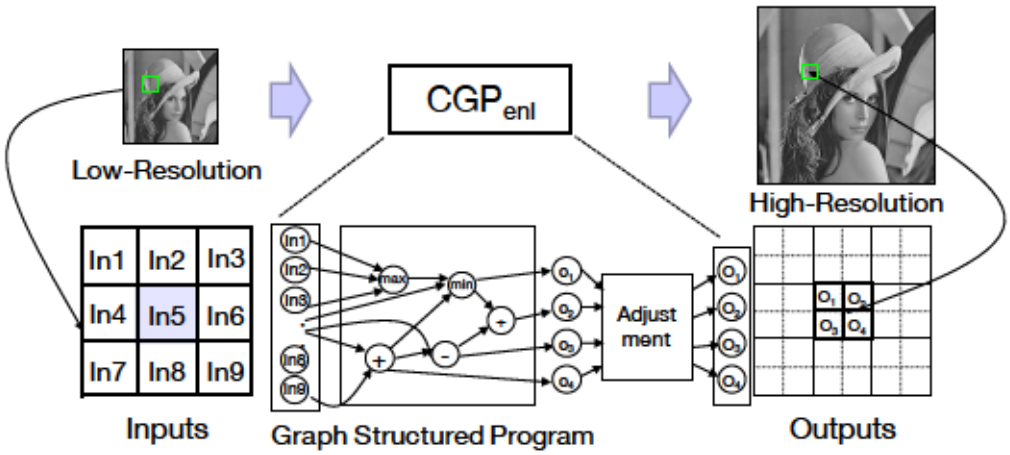

図 3.1: S=2，I=3 のときの拡大処理の概要

学習では，目標画像を $1/S$ に縮小することで，低解像度画像を作成する．作成した低解像度画像を入力として，超解像処理を行う．その出力画像と目標画像を比較し，誤差が小さくなるようにグラフ構造式の最適化を行う．

3.3 複数のグラフ構造式を組み合わせた超解像処理

3.3.1 概要と特徴

本手法には，高解像度画像を出力する拡大処理（CGP_{enl}）とその補正を行う補正処理（CGP_{up}）の 2 つがあり，それぞれの処理を入力画像の各画素ごとに行う．拡大処理だけでは，低解像度画像で失われている高周波成分のエッジやテクスチャなどを精度良く復元することは難しい．そこで，拡大処理に 1 つまたは複数の補正処理を組み合わせることで，難しい処理を分割して構築することができ，精度の向上が期待できる．

3.3.2 補正処理

拡大処理と補正処理を組み合せた処理の概要を図 3.2 に示す．補正処理は，拡大処理と同様に低解像度画像の近傍画素を入力として構築済みの拡大処理の画素値に対する補正値を出力する．出力された値は正負両方の値をとることができ，その値を拡大処理の出力に加算することで補正処理を行う．補正処理と拡大処理の出力数は同じだが，補正処理では出力値に対する調整処理は行わな

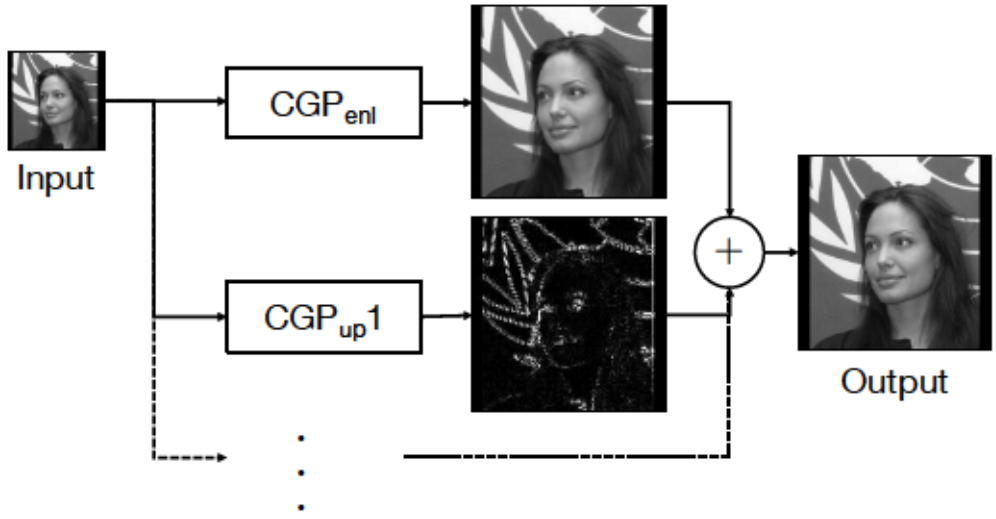

図 3.2: 拡大処理と補正処理の組み合わせた処理の概要

い．拡大処理の出力画像を入力として補正処理を行い，その出力を出力画像とする方式も考えられるが，超解像処理は，入力画像と出力画像の値に大きな変化がないため，拡大処理の出力画像を元に補正を行う方式を用いた．

3.3.3 全体の超解像処理の流れ

倍率 $S = 2$，$I = 3$ としたときの各画素における処理の流れを次に示す．

Step1 入力画像の各画素値を最大階調値 255 で除算することで [0.0，1.0] の実数値に正規化する．

Step2 画像の各画素とその近傍 3 × 3[Pixel] を CGP_{enl}，CGP_{up} に入力し演算を行う．

Step3 拡大処理の出力を式 3.1 によって 4 出力の平均と入力の中心画素が
等しくなるよう出力の値を調整する．

Step4 拡大処理の出力に補正処理の出力を加算する．

Step5 出力値が 0.0 以下の場合は 0.0 に，1.0 以上の場合は 1.0 にする Clip 処理を
行うことで出力の範囲を [0.0，1.0] にする．

Step6 出力値を 255 倍することで [0，255] の画素値を出力する．

Step7 Step1〜6 を入力画像のすべての画素に対して行う．

3.3.4 グラフ構造式の最適化

超解像処理の最適化は拡大処理，補正処理の順に 1 つずつ行う．拡大処理，補正処理の最適化には，目標画像と入力画像から構成される学習画像セットを用意する．各処理を入力画像から目標画像を出力することができるように CGP を用いて最適化を行い処理を構築する．目標画像は高解

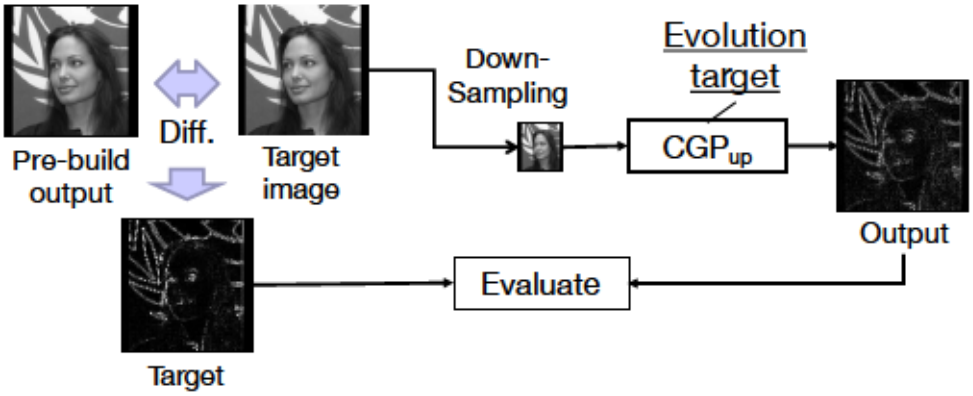

図 3.3: 補正処理の学習の概要

像度画像，入力画像は目標画像を $1/S$ に縮小した低解像度画像である．補正処理の学習の概要を図 3.3 に示す．まず拡大処理の構築を先に行い，その後，拡大処理を固定し，目標画像と構築済みの拡大処理の出力の差分が小さくなるように補正処理の構築を行う．複数の補正処理を用いる場合は，それまでに構築済みの拡大処理，補正処理を固定しその出力と目標画像の差分が小さくなるように補正処理を構築する．同時に複数の処理の構築を行うこともできるが，1 つずつ処理を構築した方がそれまでに構築された処理を元に補正処理を構築できるため，安定した最適化を行うことができると考えられる．

3.3.5 GPU を用いた並列処理による高速化

処理の高速化のために GPU を用いて処理の並列化を行う．GPU のプログラミング環境として，NVIDIA 社によって提供されている CUDA (Compute Unified Device Architecture) [42]を用いる．各画素ごとに独立して行う拡大処理，補正処理を全画素に対し並列に処理を行う．

3.4 超解像処理実験

3.4.1 実験設定

提案手法の有効性を示すため超解像処理実験を行った．画像はグレースケールとし，倍率 S を 2 倍に設定した．1/2 に縮小した画像を入力画像として超解像処理で 2 倍に拡大し，縮小前の画像を目標画像として比較することで評価を行った．4 倍，8 倍などの高倍率や任意倍率なども考えられるが，2 倍は様々な文献で使われている一般的な実験設定であるため 2 倍で評価を行った．

画質評価指標

画質評価として，Peak Signal-to-Noise Ratio（PSNR）と PSNR と比べて視覚的劣化に相関が高いといわれている SSIM を用いた．PSNR を式 3.2 に示す．

$$\mathrm{PSNR} = 10\log_{10}\frac{V_{\max}^2}{\frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}(o(x,y)-t(x,y))^2} \tag{3.2}$$

(3.3)

ここで，M，N は画像の幅と高さ，$t(x, y)$，$o(x, y)$ は，それぞれ目標画像と出力画像の (x, y) の値，V_{max} は最大階調値（255）である．

SSIM は，式 3.4 を用いて画像の 11×11 のウィンドウごとに算出し，その平均を画像の SSIM の値とする．

$$\mathrm{SSIM} = \frac{(2\mu_o\mu_t + C_1)(2\sigma_{ot} + C_2)}{(\mu_o^2 + \mu_t^2 + C_1)(\sigma_o^2 + \sigma_t^2 + C_2)} \tag{3.4}$$

ここで，μ_o, μ_t は，出力画像と目標画像のウィンドウ内の平均，$\sigma_o, \sigma_t, \sigma_{ot}$ は，出力画像と目標画像のウィンドウ内の分散と共分散，C_1，C_2 は定数であり，$C_1 = 0.01 \times V_{max}$，$C_2 = 0.03 \times V_{max}$ である．また，平均，分散はガウシアンの重み付きで算出される．

実験画像セット

実験画像は傾向の異なる 3 つの画像セットを用いた．画像セットは学習用の学習画像と未知に対する性能を検証するためのテスト画像で構成される．1 つ目は Labeled Faces in the Wild [43]から 4 枚を学習画像，24 枚をテスト画像として得た顔画像（Face），2 つ目は Caltech Buildings Dataset [44] から 4 枚を学習画像，40 枚をテスト画像として得た建物画像（Building），3 つ目は筆者らの研究室で作成した 4 枚を学習画像，4 枚をテスト画像としたテキスト画像（Text）である．画像のサイズはそれぞれ，Face は 250×250 [pixels]，Building は 512×384 [pixels]，Text は 650×350 [pixels] である．

提案手法の設定

拡大処理と補正処理の CGP のパラメータを表 3.1 に，CGP の演算ノードに用いた関数のリストを表 3.2 に示す．演算ノードには，計算コストの低い演算から選択した．表 3.2 の演算ノードでは，0.3 倍など表現することが難しい演算も存在するが，様々な値を選択肢として用意すると探索範囲が広くなり，学習が難しくなる．そのため，できるだけ少ない演算子で処理を構築した．演算子のさらなる検討は今後の課題となっている．以下，提案手法の拡大処理のみ，拡大処理と 1 つの補正処理，拡大処理と 2 つの補正処理のそれぞれの設定を Ours1，Ours2，Ours3 と呼ぶ．提案手法の処理には 5 試行の最良個体を用いる．

比較手法

比較手法として，Interpolation-based の手法から Lanczos，Example-based の手法から Glasner らの手法，進化計算法を用いた手法から CRFCN [41] を用いた．以下，Lanczos， Glasner，CRFCN

表 3.1: CGP の設定

Parameter	Value
Generation alternation model	MGG*
Number of generation	200000
Population size	50
Number of children	20
Crossover rate	1.0
Mutation rate	0.05
Input nodes	25 (5×5 [pixels])
Function nodes	40
Output nodes	4 (2×2 [pixels])
Number of CGP_{up}	2

* Minimal Generation Gap [5]

と呼ぶ．比較手法のパラメータを表 3.3 に示す．CPU で行った全ての実験と CRFCN の GPU の実験に使用した PC のスペックは，CPU : 2.5GHz，Memory : 4GB，GPU : Quadro Fx1800 である．提案手法の GPU 上での実験に使用した PC のスペックは，CPU : 2.4GHz，Memory : 4GB，GPU : Quadro 600 である．CRFCN の結果画像は，CRFCN [41]の著者から提供していただいた画像を用いた．Lanczos, Glasner らの手法の設定を 3.3 に示す．CRFCN は論文 [41]と同様の設定を用いた．CRFCN の GA のパラメータ設定を表 3.4 に示す．Glasner らの手法は，Yang ら [45]の Web ページのソースコード[1]を元に筆者らが作成したものを用いた．

適応度関数

拡大処理の適応度関数を式 3.5 に示す．

$$\text{Fitness} = 1 - \frac{\sum_{y=1}^{N}\sum_{x=1}^{M}|o(x,y) - t(x,y)|w(x,y)}{V_{\max}\sum_{y=1}^{N}\sum_{x=1}^{M}w(x,y)}. \tag{3.5}$$

ここで，$w(x, y)$ は重みの (x, y) の値である．重みにはエッジ部分を重点的に学習するため，カーネルサイズが 3×3 の Sobel フィルタのエッジ強度画像を膨張処理した画像の階調値を重みとして用いる．膨張処理では，Sobel フィルタを適用した画像の各画素の 8 近傍の最大値を求め，その最大値がしきい値 $\text{Th}_{\text{weight}}$ 以上だった場合にその中心の画素値を 8 近傍の最大値にすることで膨張処理を行う．本章では $\text{Th}_{\text{weight}} = 80$ を用いた．PSNR は 2 乗誤差を用いているので本来は評価関数も 2 乗誤差を用いるべきであるが，CRFCN の手法と評価関数を合わせるために式 3.5 を用いた．式 3.5 は絶対値誤差なので 2 乗誤差と相関が高く PSNR の値と近い評価値であるが．評価指標の検討は今後の課題である．

補正処理の最適化では，最適化を行った拡大処理の結果を入力画像，拡大処理と同様に縮小前の画像を目標画像として補正処理の最適化を行う．補正処理の最適化では，SSIM を適応度関数に用いる．

[1] http://eng.ucmerced.edu/people/cyang35

表 3.2: CGP で用いた演算子

Function	Number of input	Description
+	2	Add two input
−	2	Subtract input2 from input1
Max	4	The largest value of inputs
Min	4	The smallest value of inputs
Avg	4	The average value of inputs
*0.1	1	Multiply input by 0.1
*0.25	1	Multiply input by 0.25
*0.5	1	Multiply input by 0.5
*2.0	1	Multiply input by 2.0

表 3.3: 比較手法の設定

Methods	Parameters	Values
Lanczos	Radius of kernel r	2
Glasner	Patch size	5×5
	Resolution levels	5
	Number of nearest patches	2
	Number of iterations for back projection	3

表 3.4: CRFCN の GA のパラメータ設定

Parameter	Value
Generation alternation model	MGG*
Number of generation	20000
Population size	100
Number of children	30
Crossover rate	0.7
Mutation rate	0.05
Mutation rate (number of hidden units)	0.1

* Minimal Generation Gap [5]

3.4.2 実験結果と考察

PSNR，SSIM の画質評価の結果を表 3.6，表 3.5 に示す. なお，最良の評価値は太字で表記している．提案手法は，Lanczos と比較して PSNR，SSIM の値ともに優れている．CRFCN と比較して，Face テスト画像の PSNR の値はわずかに CRFCN が優れているが提案手法も近い値を示しており，それ以外の PSNR，SSIM の数値は提案手法が優れている．また，Glasner らの手法と比べると，全体的に PSNR，SSIM の数値はわずかに劣っているが近い値を示している．

学習画像のカテゴリと異なるテスト画像に対する画質評価結果を表 3.8 に示す．傾向の異なる画像に適用した結果では，それぞれが学習した画像セットに対する結果が最も高いものとなっており，学習に特化した処理を獲得することができている．Face，Building をそれぞれを学習した処理は，どの傾向に適用した場合においても，Lanczos の SSIM，PSNR の結果をすべて上回っている．Text を学習した処理は，Face や Building に対して，画質良く処理を行うことができていない．このことは，Face，Building と Text では画像の傾向が異なるためと考えられる．これらのことから，本手法は，ある傾向の画像セットに特化して学習することで高画質な処理を行える手法であるといえる．

また，テスト画像に対する処理時間の結果を表 3.7 に示す．CPU 上で行った実験では，実験環境は異なっているが，提案手法が最も高速に処理を行うことができている．Lanczos や CRFCN は，拡大後の高解像度画像の画素数分の処理が必要であるが，提案手法は，入力の低解像度画像の画素数分の処理が必要であり，演算ノードに単純な演算を用いているため，高速に処理を行うことができていると考えられる．GPU 上で行った実験において，提案手法は CPU 上での処理と比べて 13 倍高速に処理を行うことができており，CRFCN と比べて GPU 上でも高速に処理を行うことができている．Glasner は入力画像から低解像度と高解像度画像のペアをつくり，探索を行うため非常に時間がかかっている．

各手法の Face，Building，Text のテスト画像結果の一部をそれぞれ図 3.4（Face），図 3.5（Building），図 3.6（Text）に示す. Lanczos の結果は，すべてのテスト画像の傾向に対してぼけた画像となっている．CRFCN，Glasner，提案手法を比べると，大きな違いはなく目標画像に近い結果を出力できている．しかし，Glasner らの手法と比べて提案手法では，Face の輪郭のエッジの部分や，Building の斜めや曲線のエッジ，Text の曲線のエッジ部分などで不自然な出力になっている．

提案手法の 5 試行の学習結果について，PSNR，SSIM の最良値，平均値，標準偏差をそれぞれ表 3.9, 表 3.10 に示す．補正処理を加えていくことですべてのカテゴリにおいて，最良値，平均値ともに PSNR，SSIM の数値を改善することができている．また，SSIM の標準偏差は補正処理を加えることで小さくなっており，安定した学習が行えているといえる．

構築された拡大処理と補正処理のグラフ構造式をそれぞれ図 3.7，図 3.8，図 3.9 に示す．構築された各処理の入力ノード数は，それぞれ 18，10，17 であった．入力は，5×5 カーネルであるが，すべての入力を選択せずに計算コストを削減することができている．

これらのことから，提案手法は，Lanczos や CRFCN と比べて画質良く処理を行え，Glasner らの手法と比較して，画質評価では劣るものの目標画像に近い画像を出力できている．また，処理時間の比較では，提案手法が最も高速に処理を行うことができているおり，高速で高画質な画像を生成する超解像処理を構築できたといえる．

表 3.5: 各手法の SSIM の評価

SSIM		Lanczos	CRFCN	Glasner	Ours1	Ours2	Ours3
Training	Face	0.945	0.958	0.964	0.961	0.964	**0.965**
	Building	0.905	0.911	**0.925**	0.917	0.920	0.921
	Text	0.831	0.910	0.915	0.872	0.915	**0.922**
Test	Face	0.936	0.947	**0.955**	0.951	0.953	0.954
	Building	0.891	0.905	**0.916**	0.908	0.911	0.912
	Text	0.863	0.922	**0.935**	0.894	0.923	0.926

表 3.6: 各手法の PSNR の評価

PSNR		Lanczos	CRFCN	Glasner	Ours1	Ours2	Ours3
Training	Face	32.69	35.07	**35.79**	35.07	35.47	35.69
	Building	30.46	30.15	**31.05**	30.44	30.58	30.70
	Text	17.36	19.20	**20.23**	18.41	19.58	19.98
Test	Face	32.39	34.21	**34.85**	33.70	33.90	34.10
	Building	29.41	30.30	**31.13**	30.55	30.65	30.75
	Text	19.01	20.42	**21.72**	19.67	20.38	20.67

表 3.7: テスト画像の処理時間の比較

	CPU				GPU	
	Lanczos	CRFCN	Glasner	Ours3	CRFCN	Ours3
Face [msec/image]	132	580	728000	41	48	2.94
Building [msec/image]	432	1010	5633000	142	78	8.02
Text [msec/image]	520	2280	7573000	157	160	13.0

表 3.8: 学習とテストで画像の傾向を変えた場合の PSNR，SSIM

			Test Type					
			SSIM			PSNR		
			Face	Building	Text	Face	Building	Text
Training Type	Face	Training	**0.965**	0.914	0.856	**35.69**	30.29	17.89
		Test	**0.954**	0.907	0.883	**34.10**	30.42	19.46
	Building	Training	0.958	**0.921**	0.861	34.40	**30.70**	17.87
		Test	0.950	**0.912**	0.888	33.94	**30.75**	19.50
	Text	Training	0.830	0.808	**0.922**	27.09	25.75	**19.99**
		Test	0.843	0.811	**0.926**	27.41	26.22	**20.67**

表 3.9: Best, Average（Avg.）and Standard deviation (Std.) of SSIM

			Ours1	Ours2	Ours3
Face	Training	Best	0.962	0.964	0.965
		Avg.	0.959	0.964	0.965
		Std.	3.085×10^{-3}	0.217×10^{-3}	0.094×10^{-3}
	Test	Best	0.951	0.953	0.954
		Avg.	0.948	0.953	0.954
		Std.	3.024×10^{-3}	0.143×10^{-3}	0.160×10^{-3}
Building	Training	Best	0.917	0.920	0.921
		Avg.	0.912	0.919	0.921
		Std.	3.307×10^{-3}	0.657×10^{-3}	0.145×10^{-3}
	Test	Best	0.908	0.911	0.912
		Avg.	0.904	0.910	0.912
		Std.	3.438×10^{-3}	0.504×10^{-3}	0.099×10^{-3}
Text	Training	Best	0.872	0.916	0.922
		Avg.	0.866	0.911	0.920
		Std.	3.734×10^{-3}	2.211×10^{-3}	1.173×10^{-3}
	Test	Best	0.894	0.923	0.926
		Avg.	0.889	0.920	0.925
		Std.	3.333×10^{-3}	2.205×10^{-3}	0.618×10^{-3}

表 3.10: Best, Average（Avg.）and Standard deviation (Std.) of PSNR

			Ours1	Ours2	Ours3
Face	Training	Best	35.07	35.47	35.69
		Avg.	34.78	35.36	35.60
		Std.	0.279	0.061	0.135
	Test	Best	33.70	33.90	34.10
		Avg.	33.43	33.90	34.02
		Std.	0.028	0.021	0.119
Building	Training	Best	30.44	30.58	30.70
		Avg.	30.20	30.26	30.60
		Std.	0.169	0.325	0.134
	Test	Best	30.55	30.65	30.75
		Avg.	30.30	30.38	30.67
		Std.	0.181	0.278	0.112
Text	Training	Best	18.41	19.58	19.98
		Avg.	18.19	19.44	19.89
		Std.	0.133	0.075	0.077
	Test	Best	19.67	20.38	20.67
		Avg.	19.52	20.34	20.59
		Std.	0.102	0.094	0.046

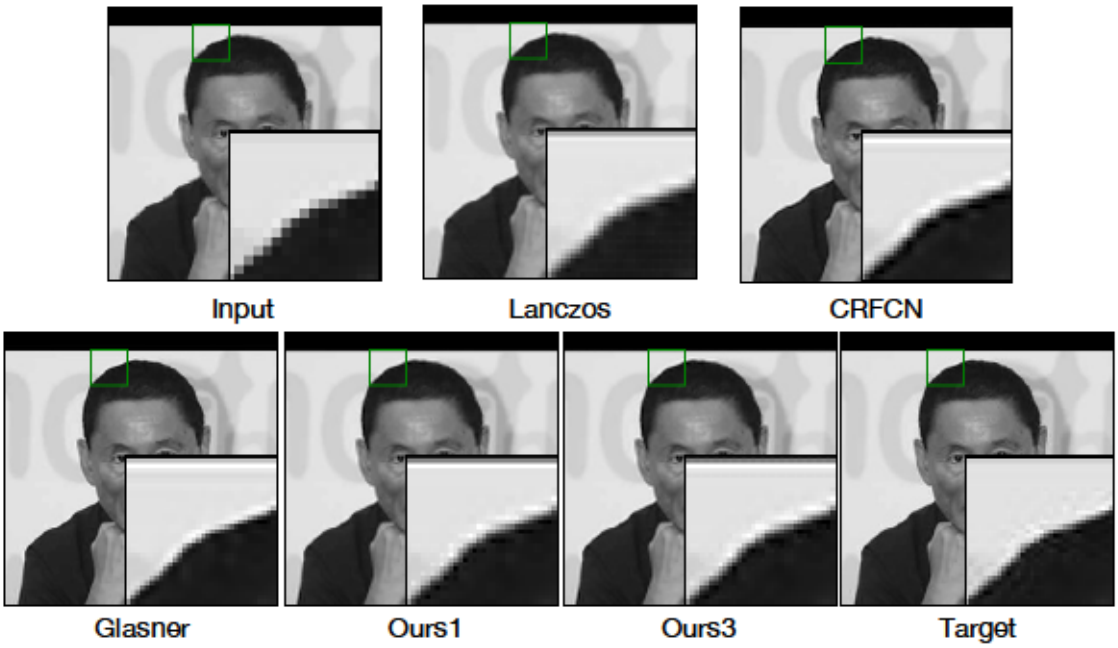

図 3.4: テスト画像の超解像処理結果（Face）

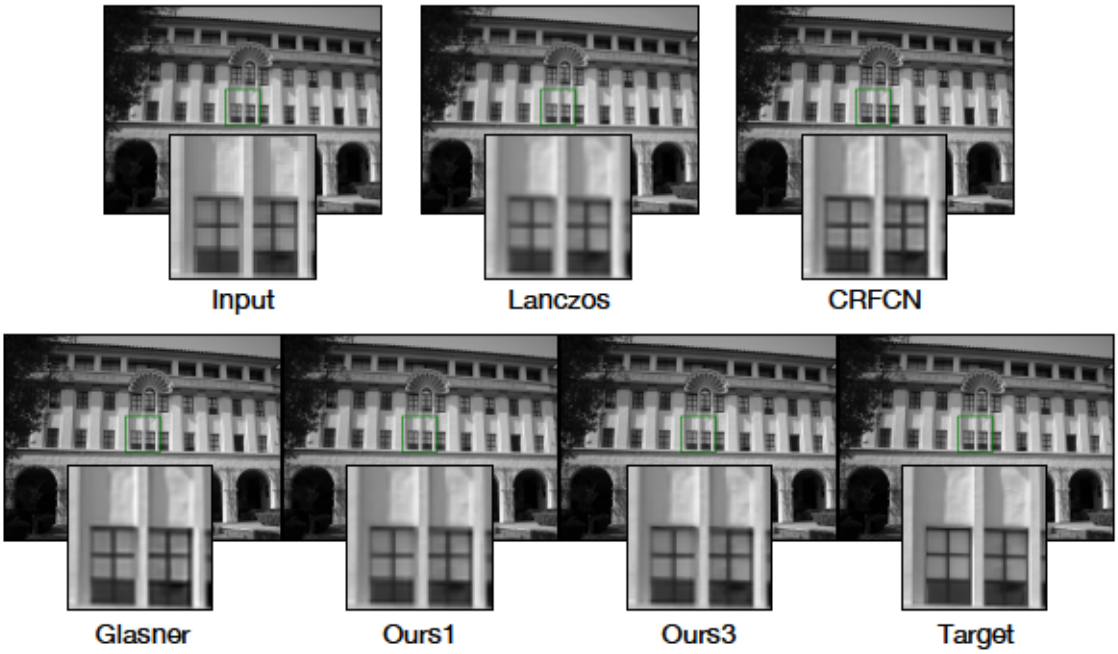

図 3.5: テスト画像の超解像処理結果（Building）

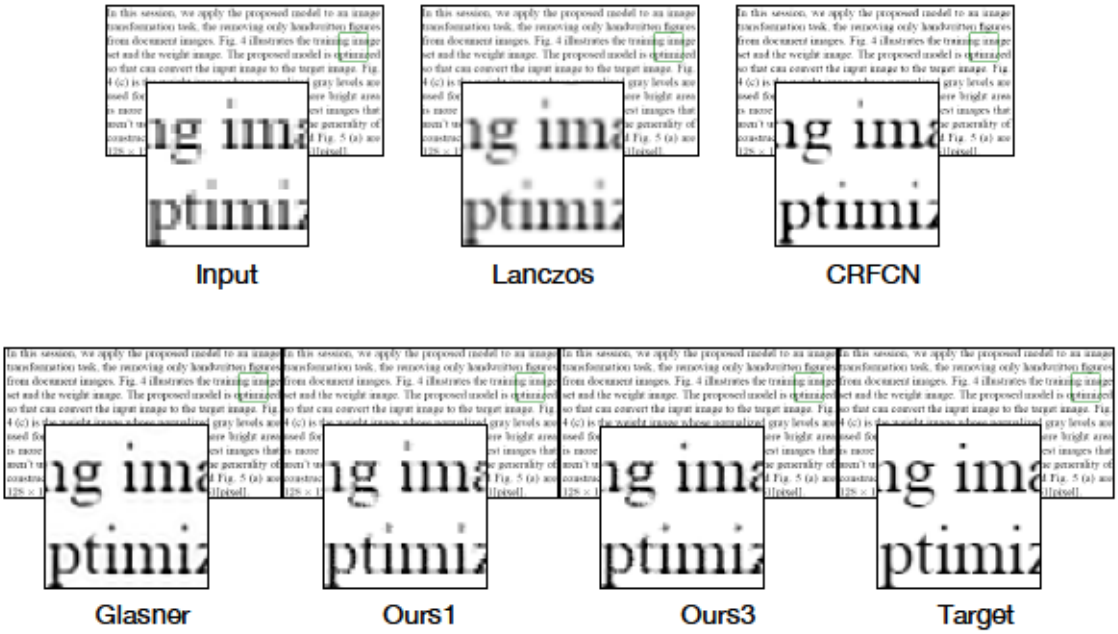

図 3.6: テスト画像の超解像処理結果（Text）

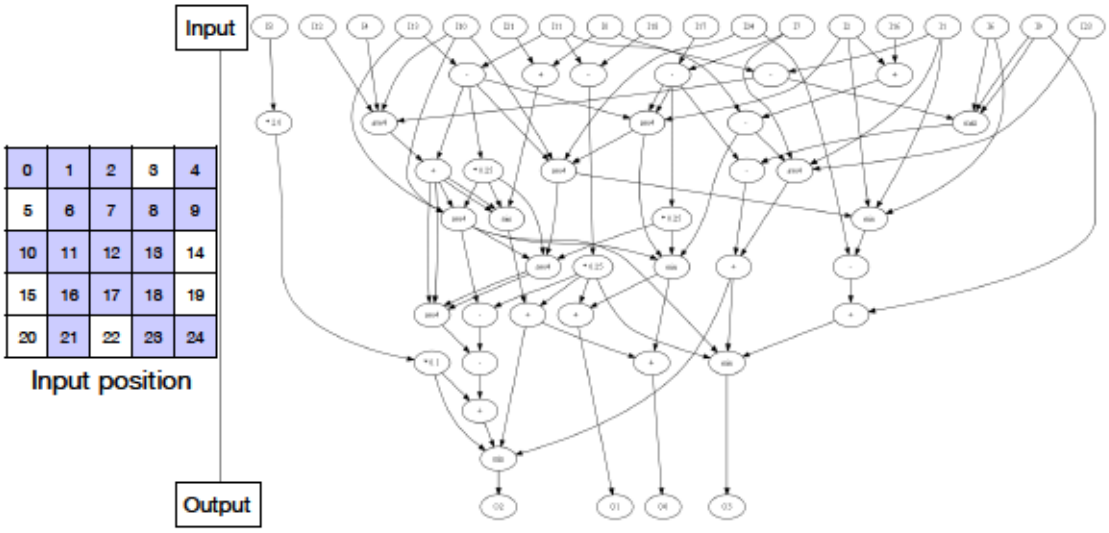

図 3.7: 構築された拡大処理（Face）

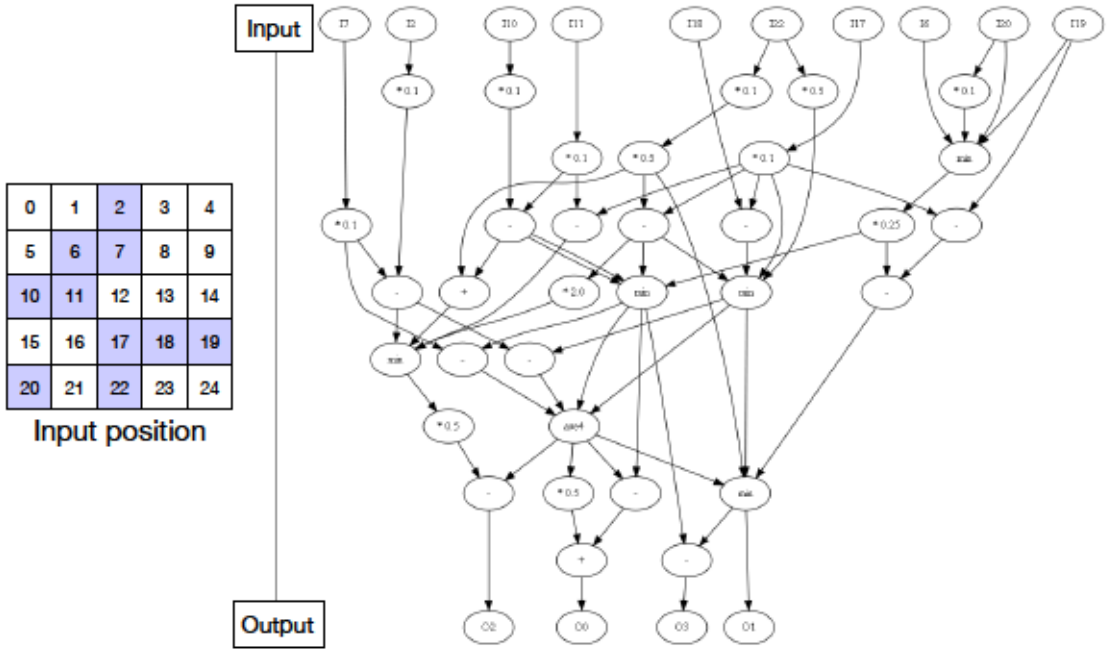

図 3.8: 構築された 1 つ目の補正処理（Face）

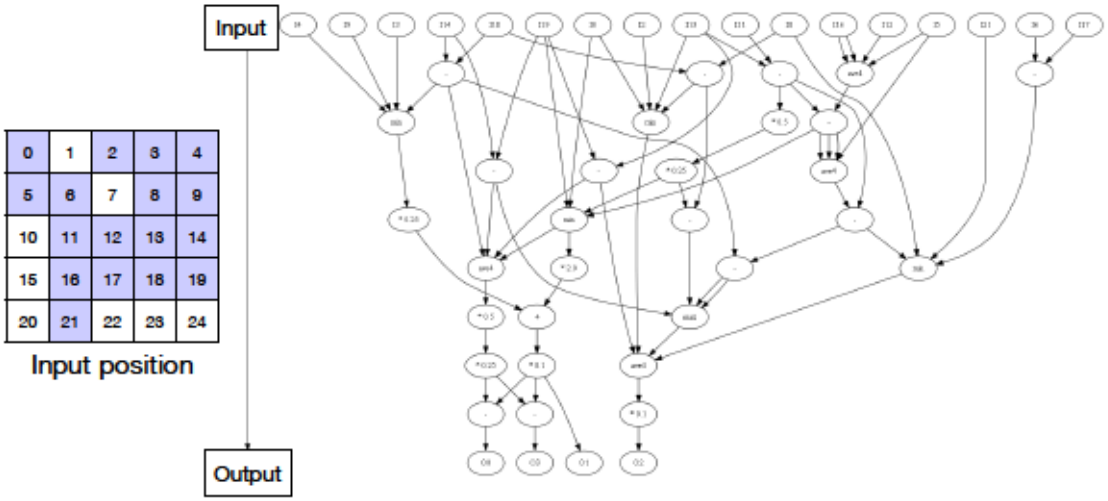

図 3.9: 構築された 2 つ目の補正処理（Face）

3.5 まとめ

本章では，進化計算法の 1 つである CGP を用いて低解像度画像の 1 つの画素およびその近傍画素の画素値から対応する高解像度画像の画素値を算出するグラフ構造式を自動構築し，シングルフレーム超解像処理を行う手法を提案した．また，複数のグラフ構造式をアンサンブルする超解像処理の構築手法を提案した．傾向の異なる顔画像，建物画像，テキスト画像の 3 つのカテゴリの画像セットに対して超解像処理実験を行い，複数のグラフ構造式をアンサンブルすることで画質が向上することを示した．従来手法との比較で，提案手法は現状の最高レベルの従来手法に近い画質の画像を高速に生成することができることを示した．

第4章　周囲の画素の関係性を考慮したシングルフレーム超解像処理の自動構築

4.1　はじめに

第3章では，拡大処理と複数のグラフ構造式をアンサンブルする手法の提案を行い，画質と計算コストの関係からその有効性を示した．
しかし，次の2つのことが課題として挙げられる．

- 入力画像の5x5の近傍入力だけから拡大処理を行っている
- 拡大画像において周囲の画素との関係が考慮されていない

入力画像の近傍入力だけでなく広い範囲を見ることでエッジやテクスチャなどを精度良く拡大することができる可能性がある．そこで本章では，拡大処理によって拡大された画像に対して，周囲の画素との関係に基づいて補正を行う補正処理を用いることで従来の手法を改良し，より高精度な超解像処理を自動構築する手法を提案する．

4.2　周囲の画素の関係性を考慮した超解像処理

4.2.1　概要と特徴

本章で提案する超解像処理の概要を図4.1に示す．本手法は，入力の低解像度画像から高解像度画像を出力する拡大処理とその出力された画像の補正を行う補正処理の2つの処理で構成される．各処理では，CGPによって構築された2つの計算式CGP_{enl}とCGP_{mod}を用いて処理が行われる．拡大処理では，低解像度画像の1つの画素およびその近傍画素の画素値から対応する高解像度画像の画素値をCGP_{enl}を用いて算出し，画像の拡大を行う．補正処理では，拡大処理によって拡大された画像からCGP_{mod}を用いて補正値を算出し，補正値を加算することで出力画像の高精度化を行う．各処理において構築する計算式はCGP_{enl}とCGP_{mod}の2つであり，それぞれの入力された画像の画素すべてに対して同じ計算式を適用する．

本手法の特徴として次の3点が挙げられる．

- 入力画像の画素値から高解像度画像の画素値を算出する拡大処理と，拡大した画像の画素値を周囲の画素との関係に基づいて補正することで出力を高精度化する補正処理の2つの処理から成り立つ．
- 学習画像に合わせてCGPを用いて各処理を自動構築することで，高精度な処理を行う．
- Example-basedの手法のように事例の探索をする必要がなく，計算コストが低い．

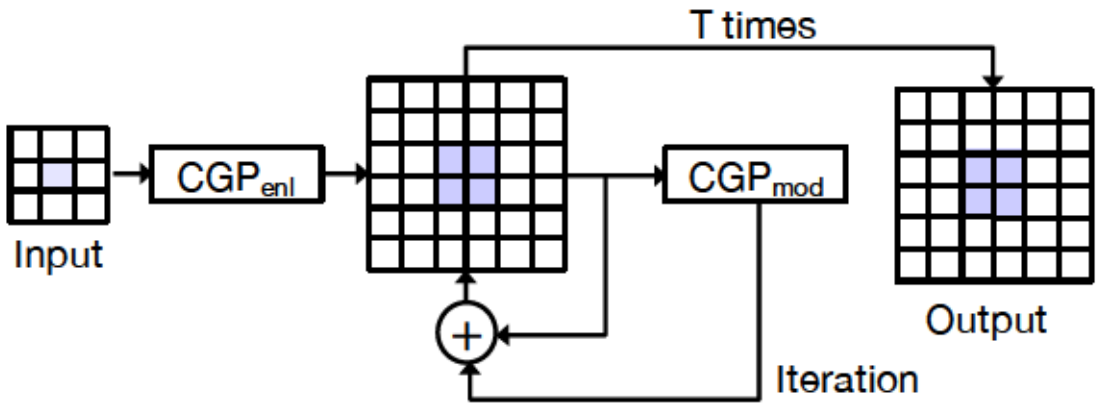

図 4.1: 提案手法の超解像処理の概要

4.2.2 補正処理

拡大処理では，拡大された各画素の境界において近傍の画素との関係性は考慮していない．補正処理では，この関係性を考慮した補正を各画素に行うことで，局所的な情報だけでは真値の推定が難しい画素の推定精度を向上させる．このように，補正処理は拡大処理によって出力された S 倍の画像を補正し，より高画質にすることを目的としている．1 つの画素に対する処理は，注目画素とその近傍画素の画素値を補正処理の $\mathrm{CGP}_{\mathrm{mod}}$ に入力し，その注目画素に対する補正値を算出する．そして，注目画素値に補正値を加算することで補正を行う．なお，補正値は正負両方の値を出力することができる．拡大した画像の各画素の周囲の情報を用いて補正を繰り返し，周囲との相互関係を利用して処理を行う．このとき繰り返し回数も $\mathrm{CGP}_{\mathrm{mod}}$ と同時に最適化される．

4.2.3 処理の流れ

超解像処理の流れを次に示す．補正処理の処理回数を T とする．

Step1 入力画像の各画素値を最大階調値 255 で除算することで [0.0，1.0] の実数値に正規化する．

Step2 入力画像の各画素値を $\mathrm{CGP}_{\mathrm{enl}}$ に入力し拡大処理を行う．その出力画像を H_0，処理回数を $t = 0$ とする．

Step3 H_t の各画素値を $\mathrm{CGP}_{\mathrm{mod}}$ に入力し補正値の算出を行う．その出力を $\mathrm{CGP}_{\mathrm{mod}}(H_t)$ とする．

Step4 H_t の各画素値に補正値を加算し，$H_{t+1} = H_t + \mathrm{CGP}_{\mathrm{mod}}(H_t)$ とする．

Step5 $t = T$ のとき Step6 へ，$t < T$ のときは $t \leftarrow t + 1$ として Step3 へ

Step6 H_T の各画素値が 0.0 以下の場合は 0.0 に，1.0 以上の場合は 1.0 にすることで出力値の範囲を [0.0，1.0] にする．H_T の各画素値を 255 倍することで [0，255] の階調画像を得る．

4.3 超解像処理の最適化

超解像処理の最適化は拡大処理，補正処理の順に行う．処理の最適化には，目標画像と入力画像から構成される学習画像セットを用意する．各処理を入力画像から目標画像を出力することができるように最適化を行い処理を構築していく．目標画像は高解像度画像，拡大処理の入力画像は目標画像を縮小した低解像度画像，補正処理の入力画像は拡大処理の出力画像である．

拡大処理では，まず高解像度画像を $1/S$ に縮小することで低解像度画像を作成する．この低解像度画像を入力画像，縮小前の高解像度画像を目標画像として，$\mathrm{CGP}_{\mathrm{enl}}$ の出力画像と目標画像の目標画像のエッジ強度の大きさを重みとした，重み付き誤差が小さくなるように最適化を行う．画像サイズを $M \times N$，目標画像と出力画像，重みの (x, y) の値をそれぞれ $t(x, y)$，$o(x, y)$，$w(x, y)$，最大階調値を $V_{\max} = 255$ とした適応度関数を式 (4.1) に示す．

$$\mathrm{Fitness} = 1 - \frac{\sum_{y=1}^{N} \sum_{x=1}^{M} |o(x,y) - t(x,y)| w(x,y)}{V_{\max} \sum_{y=1}^{N} \sum_{x=1}^{M} w(x,y)}. \tag{4.1}$$

重みにはエッジ部分を重点的に学習するため，カーネルサイズが 3×3 の Sobel フィルタのエッジ強度画像を膨張処理した画像の階調値を重みとして用いる．膨張処理では，Sobel フィルタを適用した画像の各画素の 8 近傍の最大値を求め，その最大値がしきい値 $\mathrm{Th}_{\mathrm{weight}}$ 以上だった場合にその中心の画素値を 8 近傍の最大値にすることで膨張処理を行う．

補正処理の最適化では，最適化を行った拡大処理の結果を入力画像，拡大処理と同様に縮小前の画像を目標画像として $\mathrm{CGP}_{\mathrm{mod}}$ の最適化を行う．$\mathrm{CGP}_{\mathrm{mod}}$ の最適化では，一般的に画質評価に用いられている Structural Similarity（SSIM）[46]を適応度関数に用いる．

4.3.1 GPU を用いた並列処理による高速化

本手法は，拡大処理と 1 回の補正処理は画像の各画素に対して独立に処理を行うことができるため，GPU を用いた並列処理を行うことで処理を高速化する．本手法では，GPU のプログラミング環境として NVIDIA 社によって提供されている CUDA （Compute Unified Device Architecture）[42] を用いる．

4.4 超解像処理実験

4.4.1 実験設定

提案手法の有効性を示すため，超解像処理実験を行った．画像はグレースケールとし，倍率 S を 2 倍に設定した．1/2 に縮小した画像を入力画像として超解像処理で 2 倍に拡大し，縮小前の画像を目標画像として比較することで評価を行った．画質評価として，Peak Signal-to-Noise Ratio（PSNR）と PSNR と比べて視覚的劣化に相関が高いといわれている SSIM を用いた．実験画像は 3 と同様の傾向の異なる 3 つの画像セットを用いた．画像セットは学習用の学習画像と未知に対する性能を検証するためのテスト画像で構成される．1 つ目は Labeled Faces in the Wild [43]から 4 枚を学習画像，24 枚をテスト画像として得た顔画像，2 つ目は Caltech Buildings Dataset [44]から 4 枚を学習画像，40 枚をテスト画像として得た建物画像，3 つ目は筆者らの研究室で作成した 4 枚を学習画像，4 枚をテスト画像としたテキスト画像である．それぞれ Face，Building，Text とする．画像のサイズは Face は 250×250 pixel，Building は 512×384 pixel，Text は 650×350 pixel である．

表 4.1: CGP の設定

Paramters	Values
Generation alternation model	MGG*
Crossover type	Uniform crossover
Number of generations	200000
Population size	50
Number of children	20
Crossover rate	1.0
Uniform crossover rate	0.5
Mutation rate	0.05
Input nodes	25 (5 × 5 pixel)
Function nodes	40
Output nodes	4 (2 × 2 pixel)
Functions	see Table4.2
Number of iterations	2，3，4，5

* Minimal Generation Gap [5]

CGP_{enl}，CGP_{mod} のパラメータを表 4.1 に，CGP の演算ノードに用いた関数のリストを表 4.2 に示す．CGP_{enl} のパラメータと CGP の演算ノードに用いた関数のリストは第 3 章と同様である．比較手法として，Interpolation-based の手法から Lanczos，NEDI，Reconstruction-based の手法から Sun らの手法，Example-based の手法から Freeman らの手法と Glasner らの手法を用いた．また比較として，提案手法の補正処理を行わないものも用いた（CGP_{enl}）．提案手法の処理には 5 試行の最良個体を用いる．比較手法のパラメータを表 4.3 に示す．これらのパラメータは複数のパラメータの組み合わせを試行し，学習画像に対して最も評価が高くなった値である．NEDI と Freeman らの手法はそれぞれの著者の Web ページ[1, 2]のソースコードを用いた．Sun らの手法は，Tai [21]の Web ページ[3]のソースコードを用いた．Glasner らの手法は，Yang [45]の Web ページのソースコード[4]を元に筆者らが作成したものを用いた．Lanczos と NEDI は比較手法として一般的に用いられている手法である．Sun らの手法はあらかじめエッジ強度の勾配特性を事前知識として低解像度と高解像度の画像から抽出している手法である．Freeman らの手法は提案手法と同様にあらかじめ学習画像を用いて，処理を構築している手法である．Glasner らの手法はあらかじめ学習を行わない手法であり，現状で最高レベルの手法であることから比較手法として選択した．

CPU で行った全ての実験に使用した PC のスペックは，CPU : Xeon 2.53GHz，Memory : 4GB である．提案手法の GPU 上での実験に使用した PC のスペックは，CPU : Core i7 3.40GHz，Memory : 16GB，GPU : NVIDIA GTX TITAN である．

表 4.2: CGP で用いた演算子

Functions	Number of inputs	Description
*0.1	1	$0.1 \times x$
*0.25	1	$0.25 \times x$
*0.5	1	$0.5 \times x$
*2.0	1	$2.0 \times x$
+	2	$x_1 + x_2$
-	2	$x_1 - x_2$
Max	4	$\max\{x_i,\ i = 1, 2, 3, 4\}$
Min	4	$\min\{x_i,\ i = 1, 2, 3, 4\}$
Ave	4	$\frac{1}{4}\sum_{i=1}^{4} x_i$

4.4.2 実験結果と考察

従来手法との比較

提案手法と比較手法の SSIM，PSNR の評価の結果をそれぞれ表 4.4，表 4.5 に示す．なお，最良の評価値は太字で，2 番目の評価値は下線で表記している．Lanczos，NEDI，Sun らの手法と比べて，SSIM，PSNR ともに提案手法の方が優れている．また，Freeman らの手法は学習画像に関しては SSIM，PSNR 共に最も高い値を示している．これは，Freeman らの手法は学習画像を事例として保持しているためであると考えられる．テスト画像に関しては，Freeman らの手法は大きく評価値が低くなってしまっている．これは，保持している事例と似ている部分がテスト画像に少なかったためであると考えられる．同じ学習画像から提案手法の方がより汎用的な処理を構築できているといえる．Glasner らの手法と比べると，全体的に数値はわずかに劣っているが近い値を示しており，テスト画像の Face の SSIM の値は同じ値を示している．よって本手法は，ある傾向に特化して学習することで高画質な処理を行える手法であるといえる．

各手法の画像ごとの性能依存について SSIM，PSNR の画像ごとの標準偏差，手法間の平均順位を表 4.6，表 4.7 に示す．平均順位は，SSIM，PSNR 共に平均値と全体的に同じ傾向であったが画像によっては手法の順位が変動した画像も少数見られた．提案手法の平均順位は，テスト画像において Face では最も順位が高く，それ以外の画像でも Glasner らの手法に次いだ順位だった．提案手法の標準偏差は他の手法と比較して最も小さいところもあり，それ以外のところでも他の手法と比較して大きな違いはみられなかった．このことから提案手法は，平均順位が高く安定して処理を行うことができる手法であるといえる．

テスト画像の超解像処理結果を一部切り出したものを図 4.2（Face），図 4.3（Building），図 4.4（Text）に示す．Lanczos の結果は，すべてのテスト画像の傾向に対してぼけた画像となっている．NEDI の結果は，Lanczos 同様すべてのテスト画像に対してぼけた画像となっている．Face，Building ではエッジを良好に処理できている部分もあるが，Text ではエッジ付近にノイズが出てしまっている．Sun らの手法の結果は，Face，Building ではエッジが強調された結果となっている．Text では文字が膨張されたような処理が行われている．Sun らの手法エッジの勾配特性を事前知識として処

[1] http://www.csee.wvu.edu/%7Exinl/publications.html
[2] http://people.csail.mit.edu/billf/project%20pages/sresCode/Markov%20Random%20Fields%20for%20Super-Resolution.html
[3] http://yuwing.kaist.ac.kr/projects/superresolution/index.htm
[4] http://eng.ucmerced.edu/people/cyang35

表 4.3: 比較手法の設定

Methods	Parameters	Values
Lanczos	Radius of kernel r	2
NEDI	Window size	8
	Edge pixel threshold	8
Sun	Step size τ	0.2
	Weight of gradient constraint β	0.5
	Number of iteration	20
Freeman	Low resolution patch size	7×7
	High resolution patch size	4×4
	Interval size	3
	Overlap size	2
	Number of nearest patches	20
	Number of iterations for belief propagation	10
Glasner	Patch size	5×5
	Resolution levels	5
	Number of nearest patches	2
	Number of iterations for back projection	3

表 4.4: 各手法の SSIM の評価

SSIM		Lanczos	NEDI	Sun	Freeman	Glasner	CGP_{enl}	Ours
Training	Face	0.945	0.914	0.892	**0.994**	0.964	0.961	0.965
	Building	0.900	0.866	0.873	**0.978**	0.925	0.917	0.920
	Text	0.831	0.740	0.828	**0.947**	0.915	0.872	0.906
Test	Face	0.935	0.904	0.866	0.895	**0.955**	0.951	**0.955**
	Building	0.891	0.852	0.865	0.838	**0.916**	0.908	0.911
	Text	0.863	0.788	0.858	0.845	**0.935**	0.894	0.917

理を行っているためであると考えられる．Freeman らの手法は良好に処理を行えている部分もあるがノイズが出るなど不自然な処理が行われている部分も存在した．Freeman らの手法は学習画像を保持しておき，それを用いて処理を行う手法なため学習画像と似ていない部分はうまく処理できなかったと考えられる．Glasner らの手法はすべての傾向の画像に対して鮮明でノイズが少なく，良好な結果が得られた．提案手法は，Glasner らの手法と比べると，Text などでノイズなどが見られたが Glasner らの手法に近い結果を得ることができていた．また，Lanczos，NEDI，Sun らの手法，Freeman らの手法と比べると良好に処理できているといえる．

各手法の処理時間の結果を表 4.9 に示す．NEDI，Freeman らの手法，Glasner らの手法は Matlab 上で実行した．CPU 上で行った実験では，Lanczos を除いて提案手法が最も高速に処理を行うことができた．最も画質評価が高かった Glasner らの手法は処理が複雑であり，計算コストが多くなっている．本実験において提案手法は Glasner らの手法と比較して平均で約 17,000 倍高速に処理を行えた．提案手法は，画像の局所的な入力から処理を行うことで事例の探索などをせずに処理を行なっているため，従来の手法と比べて高速に処理を行えたと考えられる．また，GPU 上で処理を行う

表 4.5: 各手法の PSNR の評価

PSNR		Lanczos	NEDI	Sun	Freeman	Glasner	CGP_{enl}	Ours
Training	Face	32.31	29.60	29.33	**39.94**	35.79	35.07	35.53
	Building	29.20	27.75	27.69	**36.26**	31.05	30.44	30.56
	Text	17.37	15.37	16.29	**22.26**	20.23	18.41	18.78
Test	Face	32.00	29.52	28.59	30.71	**34.85**	33.70	34.19
	Building	29.30	27.89	28.14	28.15	**31.13**	30.55	30.65
	Text	19.01	17.06	17.97	18.39	**21.72**	19.67	19.91

表 4.6: 各手法の画像ごとの SSIM の標準偏差と平均順位

PSNR			Lanczos	NEDI	Sun	Freeman	Glasner	CGP_{enl}	Ours
Training	Face	Std. Dev.	0.0097	0.0122	0.0175	**0.0036**	0.0067	0.0071	0.0073
		Avg. Rank	5.00	6.25	6.75	**1.00**	2.50	4.00	2.50
	Building	Std. Dev.	0.0176	0.0232	0.0226	**0.0123**	0.0158	0.0161	0.0159
		Avg. Rank	5.00	6.75	6.25	**1.00**	2.00	4.00	3.00
	Text	Std. Dev.	0.0044	0.0045	0.0035	0.0064	0.0040	0.0063	**0.0030**
		Avg. Rank	5.00	7.00	6.00	**1.00**	2.00	4.00	3.00
Test	Face	Std. Dev.	0.0241	0.0357	0.0624	0.0359	**0.0222**	0.0234	0.0227
		Avg. Rank	4.00	5.33	6.33	6.33	1.63	2.96	**1.42**
	Building	Std. Dev.	0.0287	0.0381	0.0333	0.0420	0.0271	0.0263	**0.0256**
		Avg. Rank	4.00	5.88	5.13	7.00	**1.08**	2.98	1.95
	Text	Std. Dev.	0.0431	0.0707	0.0437	0.0661	**0.0227**	0.0346	0.0251
		Avg. Rank	4.50	6.75	5.00	5.00	**1.25**	3.25	2.25

ことで CPU 上での処理と比べて，Face では約 42 倍，Building では，約 56 倍，Text では，約 70 倍の処理速度で処理を行うことができた．GPU では，画素ごとに並列に計算を行うことができるため，画素数が多い方が高速化の比率が大きくなったと考えられる．これらの結果から，提案手法は最高レベルの従来手法と比べて，近い画質で高速に超解像処理を行える手法であると考えられる．

補正処理に関する考察

構築された補正処理は Face，Building，Text それぞれ，演算ノード数が，18，19，26，処理回数が，4，2，4 であった．表 4.4，表 4.5 を見ると補正処理を行うことで学習画像，テスト画像において SSIM，PSNR 共に精度の向上を行うことができた．

構築した処理の頑健性についての考察

Face，Building，Text の画像セットの中から 2 つの画像セットを学習とし，残りの 1 つをテストとして画質の性能評価を行った．そのときの SSIM，PSNR の結果を表 4.8 に示す．テスト画像と同じ傾向の画像セットを学習した結果と比べると SSIM，PSNR の評価値は劣っていた．しかし，他の手法と比較して，SSIM では Face，Building のテストにおいて Lanczos，Glasner 以外の手法より

表 4.7: 各手法の画像ごとの PSNR の標準偏差と平均順位

PSNR			Lanczos	NEDI	Sun	Freeman	Glasner	CGP_{enl}	Ours
Training	Face	Std. Dev.	1.95	1.89	**1.05**	4.13	1.72	2.11	2.20
		Avg. Rank	5.00	6.25	6.75	**1.25**	2.00	4.00	2.75
	Building	Std. Dev.	**1.27**	1.40	1.45	2.48	1.72	1.51	1.55
		Avg. Rank	5.00	6.50	6.50	**1.00**	2.00	4.00	3.00
	Text	Std. Dev.	0.08	0.09	0.08	0.47	0.12	0.16	**0.07**
		Avg. Rank	5.00	7.00	6.00	**1.00**	2.00	4.00	3.00
Test	Face	Std. Dev.	2.38	2.36	**1.84**	2.47	2.81	2.93	3.05
		Avg. Rank	3.92	6.25	6.67	5.04	**1.29**	3.04	1.79
	Building	Std. Dev.	**1.59**	1.76	1.72	1.83	2.30	2.03	2.05
		Avg. Rank	4.00	6.58	5.78	5.65	**1.18**	2.85	1.98
	Text	Std. Dev.	1.61	1.89	1.74	1.89	1.67	1.68	**1.53**
		Avg. Rank	4.00	6.75	5.25	5.00	**1.25**	3.00	2.75

表 4.8: 3 つの画像セット（Face，Building，Text）に対するオープン評価

Training sets	Test set	SSIM	PSNR
Face, Building	Text	0.878	19.23
Building, Text	Face	0.909	29.87
Text, Face	Building	0.870	28.46

優れており，Text のテストにおいては Glasner に次ぐ評価値であった．また，PSNR では Face をテストとした PSNR 以外は，Glasner に次ぐ評価値となっていた．このことから学習画像セットの傾向と違う画像セットに適用した場合でも汎用的な処理が構築できていると考えられる．

表 4.9: CPU と GPU における超解像処理の処理時間

	Lanczos	NEDI*	Sun	Freeman*	Glasner*	Proposed (CPU)	Proposed (GPU)
Face [sec/image]	0.0110	3.98	3.11	4.81	728	0.0976	0.00232
Building [sec/image]	0.0345	12.73	11.00	15.33	5633	0.209	0.00375
Text [sec/image]	0.0423	13.21	12.66	17.28	7573	0.449	0.00638

* Executed with Matlab

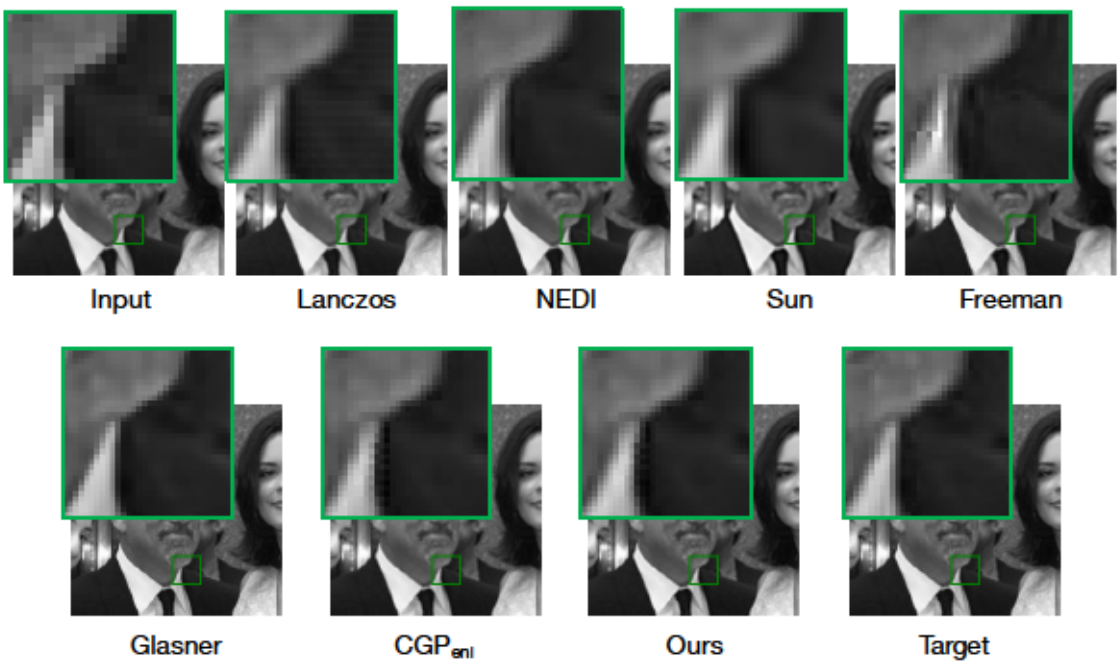

図 4.2: テスト画像の超解像処理結果（Face）

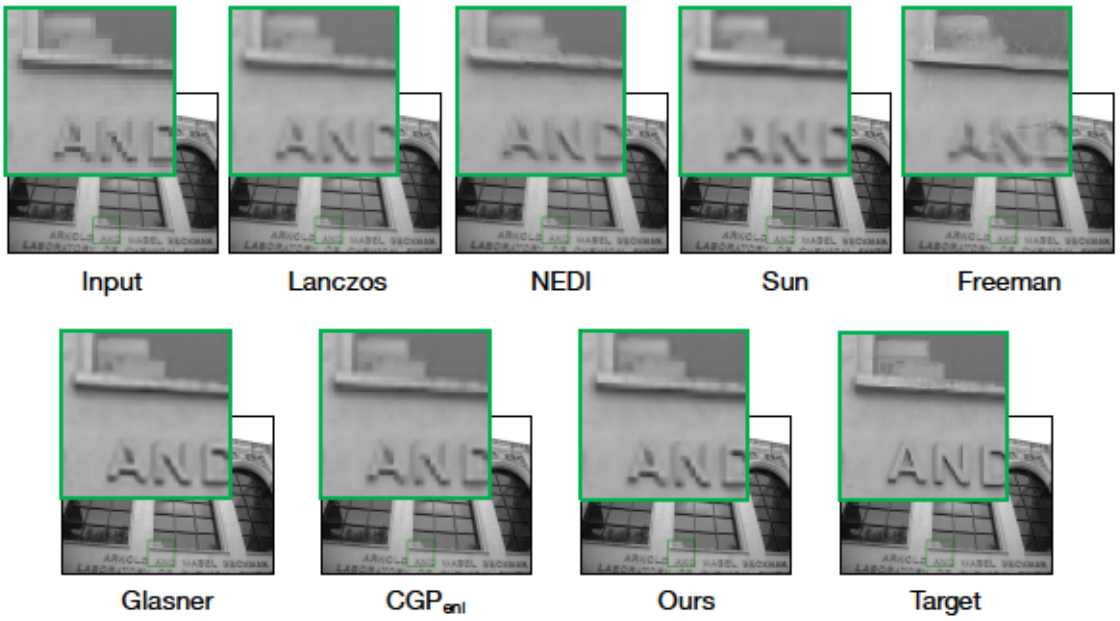

図 4.3: テスト画像の超解像処理結果（Building）

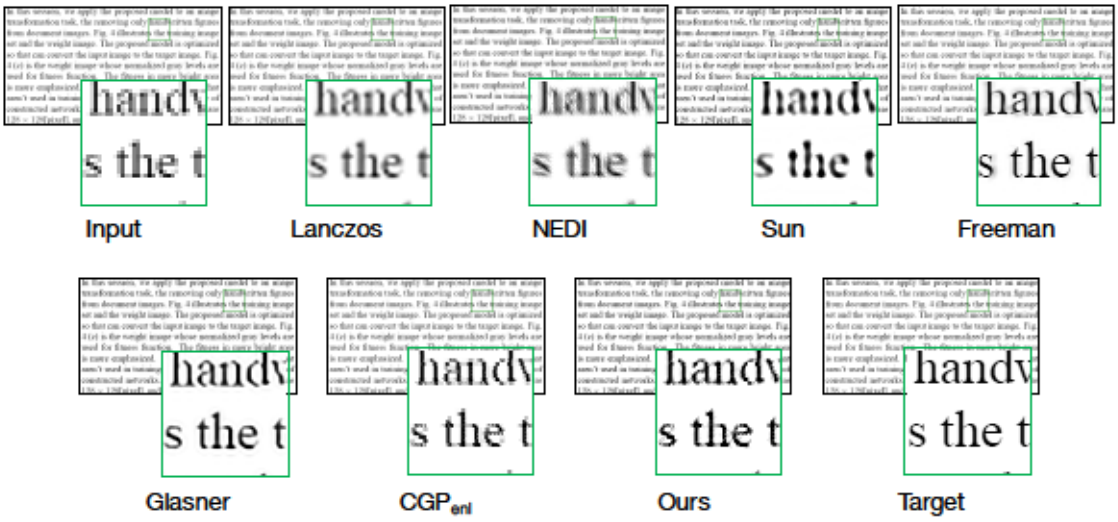

図 4.4: テスト画像の超解像処理結果（Text）

表 4.10: 各手法の PSNR の評価

PSNR		CGP_{enl}	Method1	Method2
Training	Face	35.07	35.69	35.53
	Building	30.44	30.70	30.56
	Text	18.41	19.98	18.78
Test	Face	33.70	34.10	34.19
	Building	30.55	30.75	30.65
	Text	19.67	20.67	19.9

表 4.11: 各手法の SSIM の評価

SSIM		CGP_{enl}	Method1	Method2
Training	Face	0.961	0.965	0.965
	Building	0.917	0.921	0.920
	Text	0.872	0.916	0.906
Test	Face	0.951	0.954	0.955
	Building	0.908	0.911	0.911
	Text	0.894	0.923	0.917

4.4.3 第 3 章の手法との比較

第 3 章の手法（以下，手法 1）と本提案手法（以下，手法 2）の比較を行なった．手法 1 の実験設定は，第 3 章の超解像処理実験と同様である．PSNR，SSIM の画質評価の結果を表 4.10，表 4.11 に示す．PSNR，SSIM の画質評価では，手法 1，手法 2 どちらも CGP_{enl} と比べて画質を向上することができていた．Face のテスト画像の結果では，手法 2 の方が手法 1 よりも高い PSNR，SSIM を示していた．Building のテスト画像の結果では，PSNR は手法 1 の方が高く，SSIM は同じ値であった．Text のテスト画像の結果では，手法 1 の方が手法 2 よりも高い PSNR，SSIM を示していた．これらのことから，Face 画像では，手法 2 の周囲の画素の関係性を考慮することが有効に働き，Text 画像では，手法 1 の複数のグラフ構造式をアンサンブルすることが有効に働いたと考えられる．

テスト画像の超解像処理結果を一部切り出したものを図 4.5，図 4.6，図 4.7 に示す．Face 画像では，手法 1 と比べて手法 2 はエッジ付近のリンギングノイズのような出力を抑制できている．Building 画像では，手法 2 がエッジ付近のノイズを抑制できており，目標画像に近い画像を得られている．Text 画像では，点などは手法 2 の方が目標画像に近い画像を出力できている．しかし，手法 2 は全体的にエッジ部分を濃く出力できており鮮明になっているが，目標画像とは異なってしまい，画質評価では手法 1 の方が優れていた． これらのことから，手法 2 は，手法 1 に比べてエッジ付近のノイズなどを抑制することができているといえる．

手法 1，手法 2 は，それぞれ良好に超解像処理を行えている部分と行えていない部分があるため，これらの手法を組み合わせることで精度の向上が期待できる．

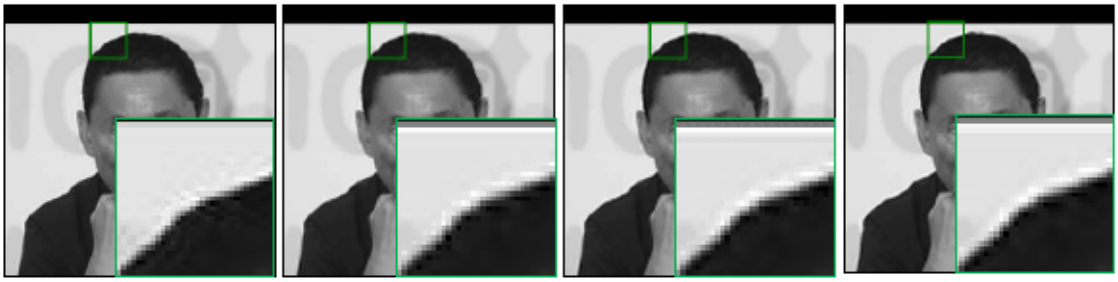

図 4.5: テスト画像の超解像処理結果（Face）

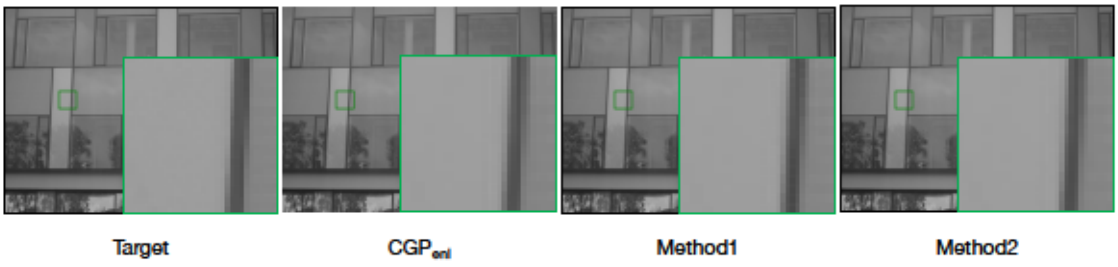

図 4.6: テスト画像の超解像処理結果（Building）

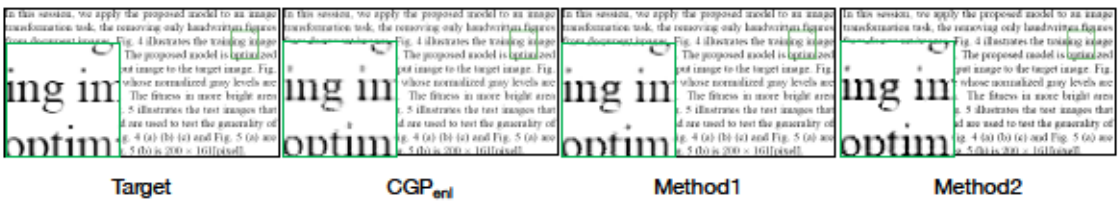

図 4.7: テスト画像の超解像処理結果（Text）

4.5 まとめ

本章では，進化計算法の 1 つである CGP を用いて低解像度画像の 1 つの画素およびその近傍画素の画素値から対応する高解像度画像の画素値を算出する処理を自動構築し，シングルフレーム超解像処理を行う手法を提案した．提案手法は，拡大処理と補正処理の 2 つで構成され，それぞれの処理を CGP を用いて構築した．顔画像，建物画像，テキスト画像の傾向の異なる 3 つの画像セットを用いて実験を行い，提案手法の有効性を検証した．従来手法との比較で，提案手法は現状の最高レベルの従来手法に近い画質の画像を高速に生成することができることを確認した．

第5章　画像の領域に適した超解像処理の自動構築

5.1　はじめに

画像の局所的な領域には，図 5.1 のようにエッジや平坦領域，テクスチャ領域など様々な領域が存在する．3 章, 4 章で提案した超解像処理手法は，すべての画素に対して同じ処理を行っており，明示的に画像の領域に適した処理を行っていない．しかし，人手によってカテゴリ分けされた画像セットに対して，超解像処理を構築し各カテゴリに特化した処理を構築することができている．そこで，入力画像の局所的な領域に対して適切な超解像処理を適用することで精度の向上が期待できる．本章では，画像の局所的な領域に特化した超解像処理を行う手法を提案する．3 章, 4 章では，カテゴリ分けした実験画像セットを用いたが，本章の超解像処理実験では，局所的な画像の領域に適した処理を行うことができているか確認するために，カテゴリ分けされていない様々な被写体が写った自然画像に対して評価を行う．

5.2　画像の領域に適した超解像処理

5.2.1　概要と特徴

本手法は，1 つの拡大処理（$\mathrm{CGP}_{\mathrm{enl}}$）と，調整処理を含まない拡大処理（$\mathrm{CGP}_{\mathrm{up}}$）と Conversion 処理（$\mathrm{CGP}_{\mathrm{conv}}$）の複数のペアから構成される．図 5.2 に本手法の概要を示す．各処理は，CGP によって構築されたグラフ構造式である．$\mathrm{CGP}_{\mathrm{enl}}$ は，入力画像を S 倍に拡大し画素値を出力することでベースとなる画像を出力する．局所的な画素ブロックの入力から S^2 個の値を出力し，調整処理を行う．$\mathrm{CGP}_{\mathrm{up}}$ は，$\mathrm{CGP}_{\mathrm{enl}}$ と同様に画素ブロックの入力から S^2 個の値を出力するが調整処理は行わずに，補正値を出力する．$\mathrm{CGP}_{\mathrm{conv}}$ は，局所的な画素ブロックの入力から 1 つの値を出力する．

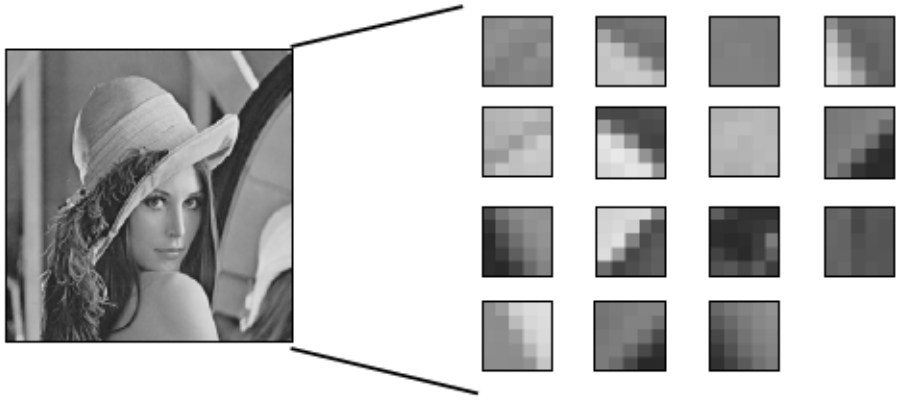

図 5.1: 画像中の局所的なパッチの例

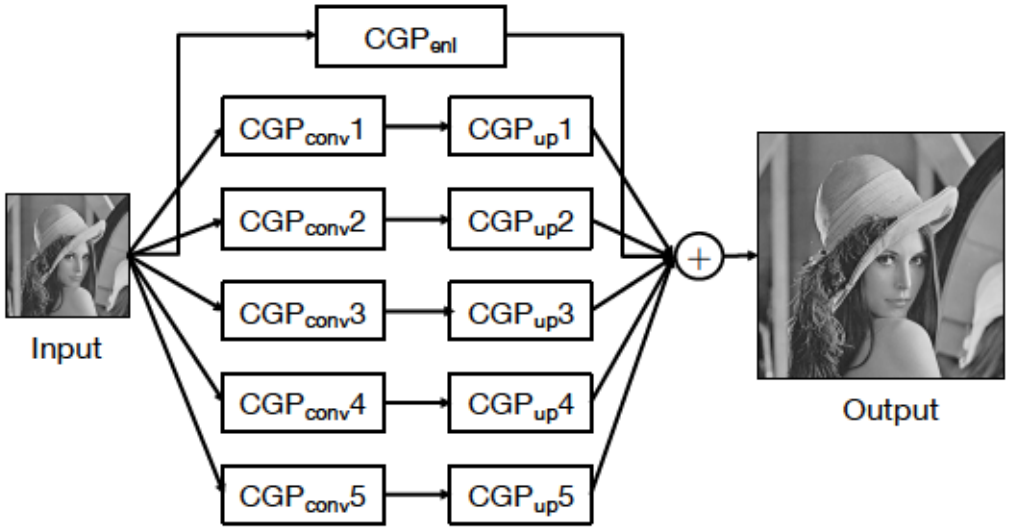

図 5.2: 提案手法の概要

この処理を入力画像のすべての画素に対して行い，入力画像と同じサイズの出力を得る．この処理は，入力画像からペアとなっている CGP$_{up}$ で処理を行う領域を抽出する．本章の超解像処理実験では，CGP$_{up}$ と CGP$_{conv}$ のペア数を 5 に設定した．

5.2.2 処理の流れ

最初に CGP$_{enl}$ で演算を行い，ベースとなる拡大画像を算出する．CGP$_{enl}$ の出力は，出力値の平均値を入力カーネルの中心の値に近づけるように各出力値を調整している．次に，CGP$_{conv}$ が入力画像を変換する．CGP$_{up}$ が変換された入力画像を拡大し，その出力を CGP$_{enl}$ の出力に加算することで最終的な出力画像を得る．

5.2.3 超解像処理の最適化

学習の流れ

各処理が異なる役割を獲得できるように学習を行うために，それぞれのグラフ構造式を同時に最適化を行う．しかし，同時に複数のグラフ構造を変化させると最適化が難しいため，世代数によって最適化対象の処理を切り替える．

学習の流れを Algorithm1 に示す．$Generation_{init}$ 世代ごとに CGP$_{enl}$，CGP$_{up}$1，CGP$_{conv}$1 の順に処理を追加していく．処理が追加されたとき，それ以前に追加された処理を固定し遺伝操作は行わない．既定数の CGP$_{up}$ と CGP$_{conv}$ のペアが追加された後は，$Generation_{repeat}$ 世代ごとに CGP$_{enl}$，CGP$_{up}$1，CGP$_{conv}$1 の順に処理の最適化を行う． 1 つの処理が最適化されているときは，それ以外の処理は固定し，遺伝操作を行わない．そして，最大世代数に到達するか適応度関数が 1.0 になるまでこの操作を繰り返す．

Algorithm 1 Training process

```
1: procedure TRAINING
2:     for i = 1 ... Generation_init do
3:         Add CGP_enl
4:         Optimize(CGP_enl)
5:     end for
6:     for j = 1 ... Number of pair do
7:         for i = 1 ... Generation_init do
8:             Add CGP_up i
9:             Optimize(CGP_up i)
10:        end for
11:        for i = 1 ... Generation_init do
12:            Add CGP_up i
13:            Optimize(CGP_conv i)
14:        end for
15:    end for
16:    while Number of iteration < Number of generation and Fitness < 1.0 do
17:        for i = 1 ... Generation_repeat do
18:            Optimize(CGP_enl)
19:        end for
20:        for j = 1 ... Number of pair do
21:            for i = 1 ... Generation_repeat do
22:                Optimize(CGP_up i)
23:            end for
24:            for i = 1 ... Generation_repeat do
25:                Optimize(CGP_conv i)
26:            end for
27:        end for
28:    end while
29: end procedure
```

適応度関数

適応度関数を式 5.1 に示す．

$$\text{Fitness} = 1 - \frac{\sum_{y=1}^{N}\sum_{x=1}^{M}|o(x,y)-t(x,y)|w(x,y)}{V_{\max}\sum_{y=1}^{N}\sum_{x=1}^{M}w(x,y)}. \tag{5.1}$$

ここで，M，N は画像の幅と高さ，$t(x,y)$，$o(x,y)$，$w(x,y)$ は，それぞれ目標画像と出力画像，重みの (x,y) の値，V_{max} は最大階調値（255）である．重みにはエッジ部分を重点的に学習するため，カーネルサイズが 3×3 の Sobel フィルタのエッジ強度画像を膨張処理した画像の階調値を重みとして用いる．膨張処理では，Sobel フィルタを適用した画像の各画素の 8 近傍の最大値を求め，その最大値がしきい値 $\text{Th}_{\text{weight}}$ 以上だった場合にその中心の画素値を 8 近傍の最大値にすることで膨張処理を行う．本章では $\text{Th}_{\text{weight}} = 80$ を用いた．

5.3 GPU を用いた並列処理による高速化

本手法で用いている CGP_{enl}，CGP_{up}，CGP_{conv} は画像の各画素に対して独立に処理を行うことができるため，GPU を用いた並列処理を行うことで処理を高速化する．本手法では，GPU のプログラミング環境として NVIDIA 社によって提供されている CUDA （Compute Unified Device Architecture）[42]を用いる．

5.4 超解像処理実験

5.4.1 実験設定

提案手法の有効性を示すために超解像処理実験を行った．画像はグレースケールとし，倍率 S を 2 倍に設定した．実験では，画像サイズを 1/2 に縮小した画像を入力画像として超解像処理で 2 倍に拡大し，縮小前の画像を目標画像として比較することで評価を行った．画質評価指標として，Peak Signal-to-Noise Ratio（PSNR）と PSNR と比べて視覚的劣化に相関が高いといわれている Structural Similarity（SSIM）を用いた．

実験画像として Lenna 画像と Berkeley Segmentation Dataset（BSDS）[1]から 3 枚を学習画像，それ以外の BSDS の 36 枚をテスト画像とした．

$Generation_{\text{init}}$ を 10000，$Generation_{\text{repeat}}$ に 500 に設定した．世代交代モデルとして初期収束を抑制し，多様性を保つ効果がある Minimal Generation Gap（MGG）を用いた．表 5.2 に CGP のパラメータを表 5.1 に CGP の演算ノードに用いた関数のリストを示す．

比較手法として，Interpolation-based の手法から Lanczos，Reconstruction-based の手法から Sun らの手法，Example-based の手法から Glasner らの手法を用いた．Glasner らの手法はあらかじめ学習を行わない手法であり，現状で最高レベルの手法である．比較手法のパラメータを表 5.3 に示す．Sun らの手法は，Tai [21]の Web ページ[2]のソースコードを用いた．Glasner らの手法は，Yang [45]の Web ページのソースコード[3]を元に筆者らが作成したものを用いた．また，200000 世代の学習を行なった CGP_{enl} を比較として用いた．

[1] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
[2] http://yuwing.kaist.ac.kr/projects/superresolution/index.htm
[3] http://eng.ucmerced.edu/people/cyang35

CPU で行った全ての実験に使用した PC のスペックは，CPU : Xeon 2.6GHz，Memory : 32GB である．GPU での実験に使用した PC のスペックは，CPU : Core i7 3.40GHz，Memory : 16GB，GPU : NVIDIA GTX TITAN である．

表 5.1: CGP の設定

Parameter	Value
Generation alternation model	MGG*
Number of generation	600000
Population size	50
Number of children	20
Crossover rate	1.0
Mutation rate	0.03

* Minimal Generation Gap [5]

表 5.2: CGP で用いた演算子

Function	Number of input	Description
+	2	Add two input
−	2	Subtract input2 from input1
Max	2	The largest value of inputs
Min	2	The smallest value of inputs
Ave	2	The average value of inputs
Abs	1	The absolute value of input
*0.1	1	Multiply input by 0.1
*0.25	1	Multiply input by 0.25
*0.5	1	Multiply input by 0.5
*2.0	1	Multiply input by 2.0

表 5.3: 比較手法の設定

Methods	Parameters	Values
Lanczos	Radius of kernel r	2
Sun	Step size τ	0.2
	Weight of gradient constraint β	0.5
	Number of iteration	20
Glasner	Patch size	5×5
	Resolution levels	5
	Number of nearest patches	2
	Number of iterations for back projection	3

5.4.2 実験結果と考察

従来手法との比較

PSNR，SSIM の画質評価の結果を表 5.4，表 5.5 に示す．各手法のテスト画像結果の一部をそれぞれ図 5.3，図 5.4，図 5.5 に示す．Lanczos の結果画像は，エッジ部分などが全体的にぼけた結果となっている．Sun の結果画像は，Zebra 画像の縞模様のエッジ部分は良好に処理をできているが，それ以外の画像ではぼけた結果となっている．Reconstruction-base の手法は，事前知識に依存するため特定のパターン以外は良好に処理を行うことができない．そのため，PSNR，SSIM の値も低くなっている．Glanser は，全体的にノイズなどなく目標画像に近い画像を生成できている．提案手法は，画質評価では Glasner の次に良い画質であった．また結果画像から鮮明性では少し劣るが Glasner と同等な出力画像を生成することができている．

また，テスト画像に対する処理時間の結果を表 5.6 に示す．提案手法は CPU 上で Lanczos，CGP_{enl} の次に高速であった．提案手法は並列処 理に適しており GPU 上では CPU と比べて 81 倍高速に超解像処理を行うことができた．提案手法は CGP_{enl} と比較して結果画像ではエッジをよりシャープにすることができており，評価値も上回っている．これらのことから複数のグラフ構造式を組み合わせることの有効性を示すことができた．

表 5.4: 各手法の PSNR の評価

PSNR（dB）	Lanczos	Sun	Glasner	CGP_{enl}	Ours
Training	29.79	28.09	**32.68**	31.43	31.96
Test	28.76	27.40	**30.87**	30.15	30.34

表 5.5: 各手法の SSIM の評価

SSIM	Lanczos	Sun	Glasner	CGP_{enl}	Ours
Training	0.924	0.892	**0.949**	0.937	0.940
Test	0.870	0.839	**0.899**	0.890	0.892

表 5.6: テスト画像の処理時間の比較（sec）

	Lanczos	Sun	Glasner*	CGP_{enl}	Ours
CPU	**0.0221**	11.58	1492	0.0441	0.2894
GPU	-	-	-	**0.001245**	0.003566

* Executed with MATLAB

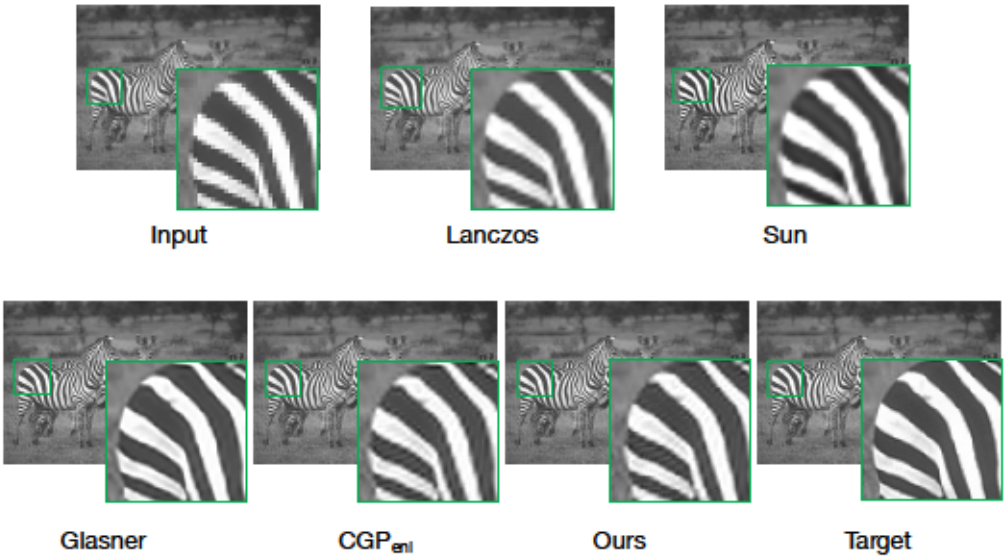

図 5.3: テスト画像結果の一部（Zebra）

図 5.4: テスト画像結果の一部（Wall man）

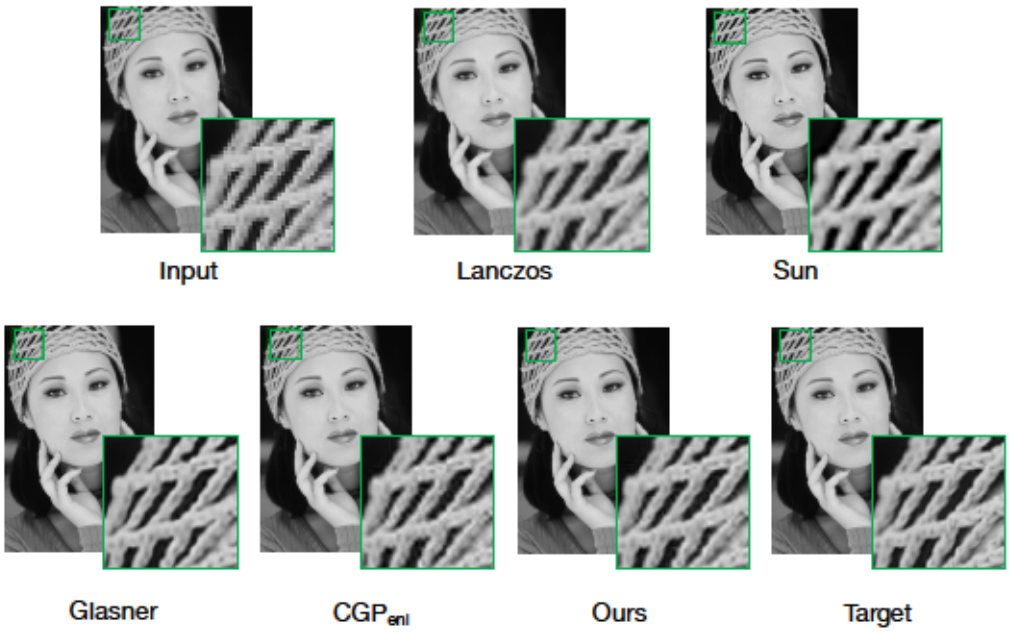

図 5.5: テスト画像結果の一部（Woman）

構築されたグラフ構造式に関する考察

テスト画像に対する CGP_{enl} と 5 つの CGP_{up} の出力を図 5.6 に示す．CGP_{up} の出力は，$[0, 255]$ の整数に正規化し画像として表示している．CGP_{enl} の出力画像では，エッジ部分がぼけており不自然な出力になっているが，提案手法の出力画像は，良好にエッジ部分を出力することができている．

CGP_{up} の出力について，$CGP_{up}1$，$CGP_{up}5$ は右下方向のエッジ，$CGP_{up}2$ は垂直方向のエッジ，$CGP_{up}3$ は水平方向のエッジ，$CGP_{up}4$ は右上方向のエッジを出力する傾向となっている．また，各処理の 5×5 カーネルの入力位置を図 5.7 に示す．CGP_{enl} は，入力カーネルの中心が選択されており，各 CGP_{up} は，それぞれ異なった傾向の入力位置を選択している．CGP_{enl} は，このように，CGP_{conv} と CGP_{up} のペアはそれぞれ異なった役割を獲得することができている．

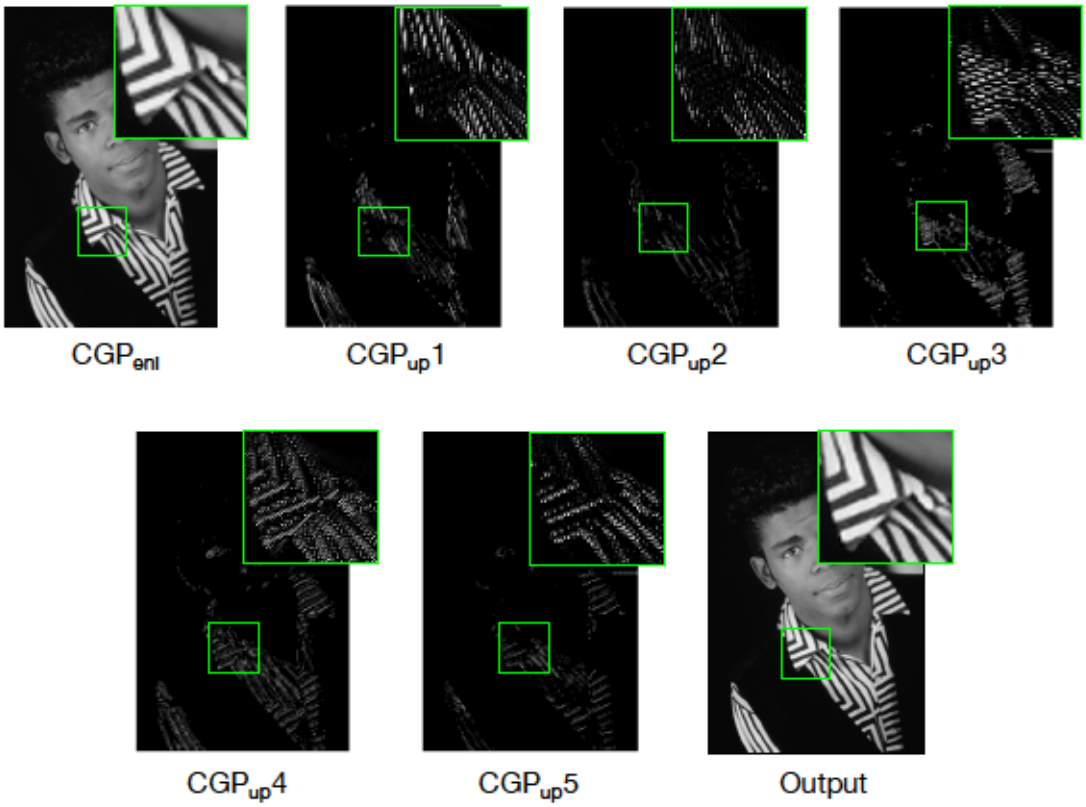

図 5.6: テスト画像に対する各処理の出力結果

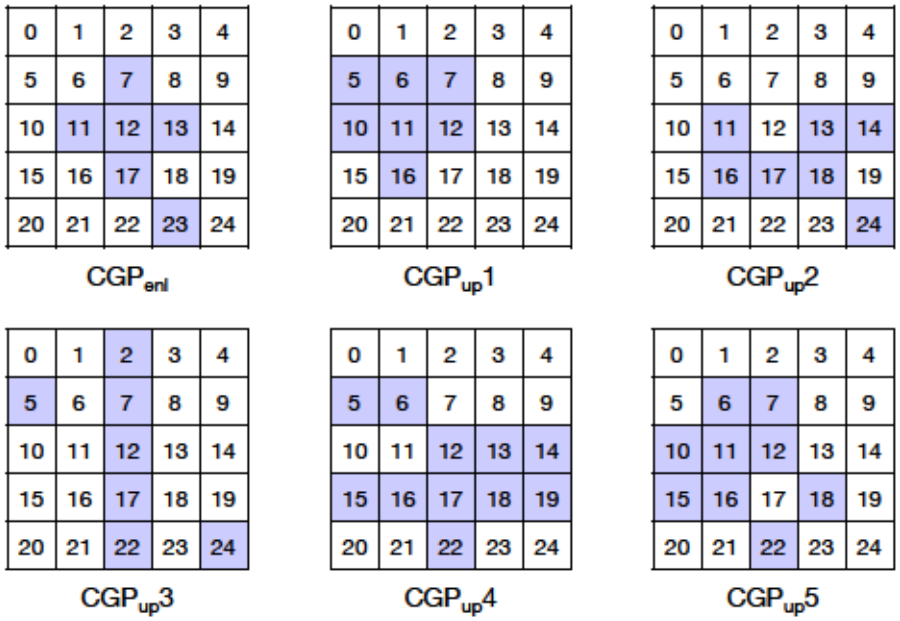

図 5.7: 各処理の 5×5 カーネルの入力位置

表 5.7: 各手法の PSNR の評価

PSNR	CGP_{enl}	Method2	Method3
Training	31.43	31.71	31.96
Test	30.15	30.31	30.34

表 5.8: 各手法の SSIM の評価

SSIM	CGP_{enl}	Method2	Method3
Training	0.937	0.942	0.940
Test	0.890	0.893	0.892

5.4.3 第 4 章の手法との比較

第 4 章の手法（以下，手法２）と本提案手法（以下，手法３）の比較を行なった．手法２の実験設定は，第 3 章の超解像処理実験と同様である．PSNR，SSIM の画質評価の結果を表 5.7，表 5.8 に示す．PSNR，SSIM の画質評価では，手法２，手法３どちらも CGP_{enl} と比べて画質を向上することができていた．PSNR の評価値では，手法３の方が手法２より学習，テスト画像共に優れていた．SSIM の評価値では，手法２の方が手法３より学習，テスト画像共にわずかに優れていた．このことは，手法２で適応度関数に SSIM を用いており，手法 1 では適応度関数に重み付きの絶対値誤差を用いているためであると考えられる．

テスト画像の超解像処理結果を一部切り出したものを図 5.8，図 5.9 に示す．図 5.8 の Man 画像では，上部の Crop 画像において，手法２では黒の線に濃淡の縞模様が出ているが，手法３では良好に処理できている．下部の Crop 画像において，手法３にはエッジの周囲にノイズが出ているが，手法２にはノイズは見られず良好に処理できている，図 5.9 の Building 画像では，上部の Crop 画像において，手法２では，斜めのエッジの付近にノイズが出ているが，手法３にはノイズは見られず良好に処理できている．下部の Crop 画像において，手法２の方が縦や斜めの枠線のエッジがシャープになっている．

手法２，手法３は，それぞれ良好に超解像処理を行えている部分と行えていない部分があるため，これらの手法を組み合わせることで精度の向上が期待できる．

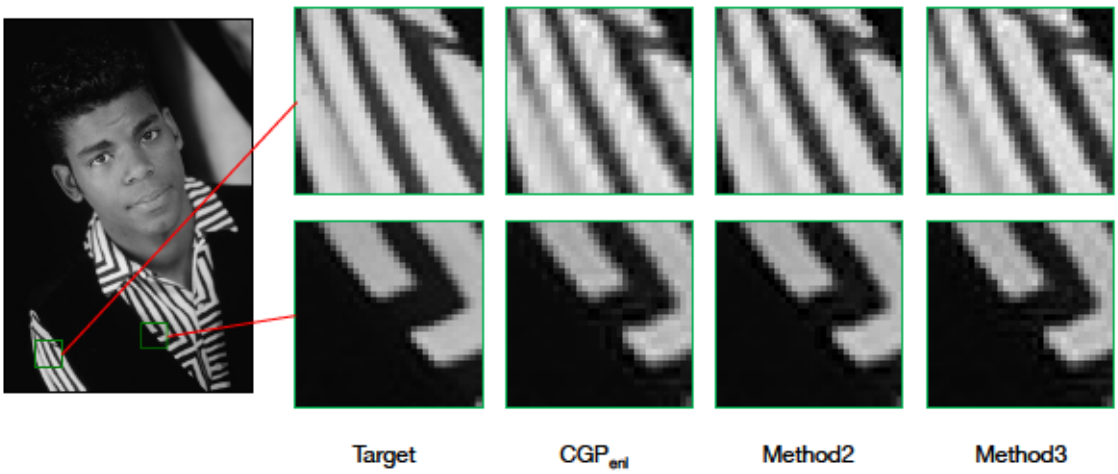

図 5.8: テスト画像の超解像処理結果（Man）

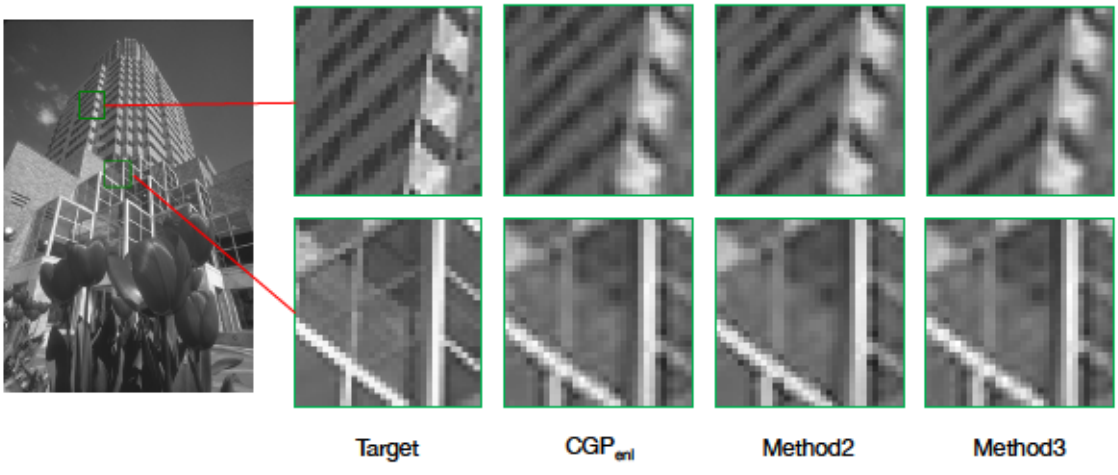

図 5.9: テスト画像の超解像処理結果（Building）

5.5 まとめ

本章では，複数のグラフ構造式を用いたシングルフレーム超解像処理を自動構築する手法を提案した．複数のグラフ構造式を世代数によって切り替えながら最適化することで，それぞれ異なった役割の処理を構築することができることを示した．また，入力画像の領域ごとに適した複数のグラフ構造式を構築することで画質が向上することを確認し，その有効性を示した．提案手法は，最高レベルの手法と比較して近い画質評価値を示しており，高速に超解像処理を行えていることから高速かつ高画質な超解像処理を行える手法であるといえる．

第6章　提案手法の組み合わせとDeep Learningを用いた手法との比較

6.1　はじめに

第3章，第4章，第5章では，それぞれ複数のグラフ構造式を組み合わせた超解像処理（以下，手法1），周囲の画素の関係性を考慮したシングルフレーム超解像処理（以下，手法2），画像の領域に適した超解像処理（以下，手法3）の3つを提案してきた．手法1では，複数のグラフ構造式を組み合わせることで精度が向上することを示した．また．手法2では，手法1で拡大画像の近傍の周囲の画素の関係性を考慮し，広い範囲を見ることができる補正処理を追加し精度が向上することを示した．手法3では，手法1，手法2がすべての画素に対して同じ処理を行っているのに対して．画像の局所的な領域に適した超解像処理を行うことで精度が向上することを示した．これら3つの手法を組み合わせることで精度の向上を期待することができる．そこで本章では，手法1と手法2の組み合わせ，手法2と手法3を組み合わせた処理を構築し，その評価を行う．

また，これまでの実験設定では少ない学習画像から学習を行っており，大量の画像セットから学習を行うDeep Learningの手法とは比較を行うことができなかった．本章では実験設定をDongらによって提案されたDeep Learningを用いた超解像処理手法であるSRCNN［30］に合わせ，超解像処理実験を行う．

6.2　提案手法の組み合わせとDeep Learningを用いた手法との比較

6.2.1　概要と特徴

手法２の補正処理は拡大画像に対して適用する処理である．繰り返し補正を行うことで，広い範囲の入力情報や周囲の画素との関係性を考慮することができる．そのため，手法２を手法１と手法３の出力画像に対して適用することで，手法１，手法３では課題となっていた拡大画像の周囲の画素の関係性を考慮することができる．これによって手法１と手法２，手法３と手法２をそれぞれ組み合わせることで精度の向上が期待できる．

手法１と手法２の組み合わせた手法（手法１＋２）の概要を図6.1に示す．手法１の複数のグラフ構造式を加算した出力に対して手法２の補正処理を適用する．本章では，第3章と同様に手法１の加算するグラフ構造式の数を３に設定した．

手法３と手法２の組み合わせた手法（手法３＋２）の概要を図6.2に示す．手法３のCGP_{enl}とCGP_{conv} - CGP_{up}のペアの出力を加算した出力に対して手法２の補正処理を適用する．本章では，第5章と同様にCGP_{conv} - CGP_{up}のペアの数を５に設定した．

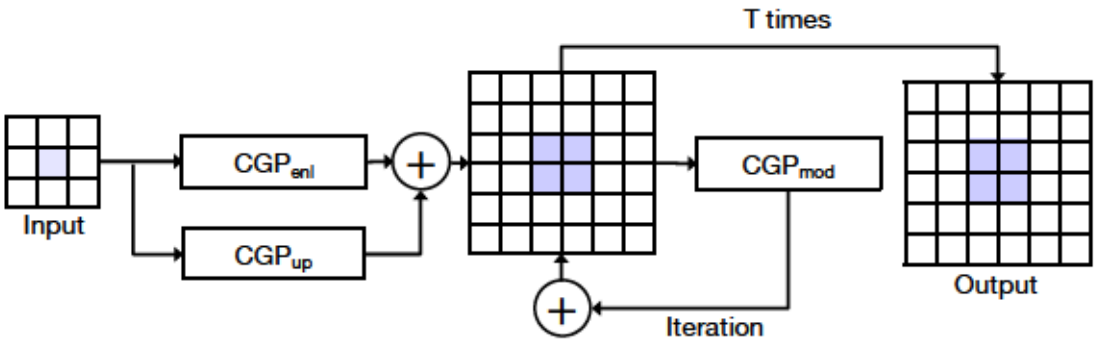

図 6.1: 手法 1 と手法 2 の組み合わせ

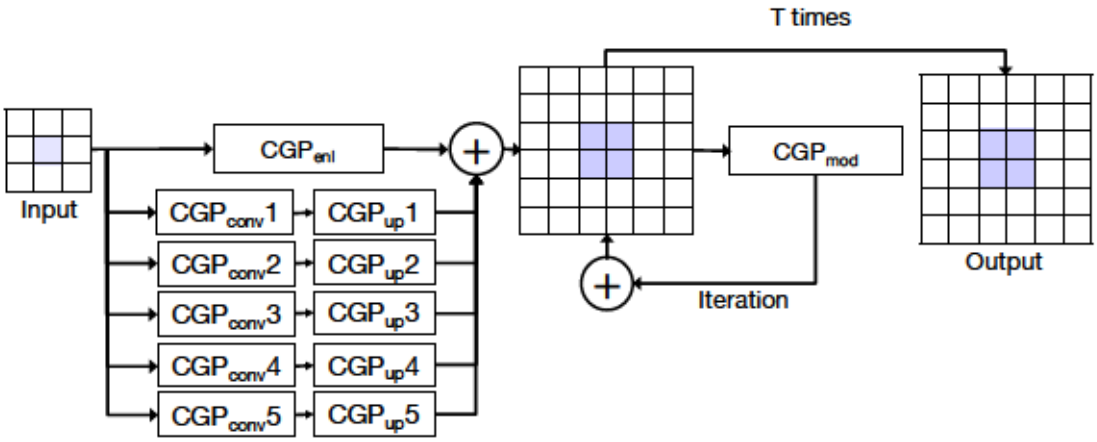

図 6.2: 手法 3 と手法 2 の組み合わせ

6.2.2 学習画像のサンプリング

画像中には様々な画素ブロックが存在する．それらに対して精度の良く超解像処理を行うためには，大量の画像から学習を行うことが考えられる．しかし，CGP を用いてグラフ構造式を構築するときには，1 個体の評価にすべての学習画像に対して適用する必要があり，計算コストが膨大になってしまう．そこで，グラフ構造式の最適化の際に，入力画像である低解像度画像から学習画素のサンプリングを行う．学習画素のサンプリング方法として，画像のエッジ部分を重点的に学習するために，カーネルサイズが 3×3 の Sobel フィルタのエッジ強度で重み付けをしたサンプリングを行い学習画素を選択する．サンプリングを行うことの利点として，処理を構築する計算コストの節約や過学習の抑制が期待できることが挙げられる．

6.2.3 超解像処理の最適化

手法 1 と手法 2 を組み合わせた手法

手法 1，手法 2 の順に処理の構築を行う．まず，第 3 章の構築法と同様に手法 1 の構築を行う．このとき入力画像が低解像度画像，目標画像が高解像度画像である．そして，構築した手法 1 部分を固定し，その出力画像を入力，高解像度画像を目標として第 4 章の構築法と同様に手法 2 の補正処理の構築を行う．

手法 2 と手法 3 を組み合わせた手法

手法 3，手法 2 の順に処理の構築を行う．まず，第 5 章の構築法と同様に手法 3 の構築を行う．このとき入力画像が低解像度画像，目標画像が高解像度画像である．また，本章では手法 3 の $\mathrm{CGP}_{\mathrm{enl}}$ は学習済みのグラフ構造式を初期値として用いた．そして，構築した手法 3 部分を固定し，その出力画像を入力，高解像度画像を目標として第 4 章の構築法と同様に手法 2 の補正処理の構築を行う．

適応度関数

適応度関数を式 6.1 に示す．

$$\mathrm{Fitness} = 1 - \frac{\sum_{n=1}^{N}\sum_{y=1}^{P_y}\sum_{x=1}^{P_x}|o(x,y)-t(x,y)|}{V_{\max}\times N\times P_y\times P_x}. \tag{6.1}$$

ここで，N はサンプリング数，P_y，P_x はサンプリングした出力パッチ幅と高さ，$t(x, y)$，$o(x, y)$ は，それぞれ目標パッチと出力パッチの (x, y) の値，V_{max} は最大階調値（255）である．

6.3 超解像処理実験

6.3.1 実験設定

本手法の有効性と Deep Learning を用いた手法と比較するために，超解像処理実験を行なった．SRCNN と実験設定を合わせて実験を行った．画像はグレースケールとし，倍率 S を 2 倍に設定した．1/2 に縮小した画像を入力画像として超解像処理で 2 倍に拡大し，縮小前の画像を目標画像として比較することで評価を行った．画質評価として，Peak Signal-to-Noise Ratio（PSNR）を用いた．

SRCNN の論文と同様に YCbCr 空間の Y 成分について画質評価を行なった．SRCNN は，筆者らの Web ページのソースコード[1]を用いた．

CGP の演算子とその積和演算数を表 6.2 に示す．比較として手法 1，手法 2，手法 3 を用いた．手法 1 + 2，手法 3 + 2 の $\mathrm{CGP}_{\mathrm{mod}}$ の繰り返し回数は 2 に固定し，手法 2 の $\mathrm{CGP}_{\mathrm{mod}}$ の繰り返し回数は 3 に固定した．提案手法の学習では，Yang らの文献 [27] で学習に用いられている 91 枚の画像を学習セットとしてサンプリングを行なった．$\mathrm{CGP}_{\mathrm{enl}}$ の学習ではサンプリング数を 200000 に設定し，それ以外のグラフ構造式の学習ではサンプリング数は 40000 に設定した．SRCNN の学習では，学習画像数が多いほど性能が高いことが示されているため，この 91 枚の学習画像セットよりも性能が高かった ILSVRC 2013 ImageNet [47] を学習画像としている．ImageNet は 395909 枚の画像から構成される物体検出の画像セットであり，提案手法では学習に計算コストが掛かるため ImageNet を使用していない．テスト画像は，Set5 と Set14 の 2 つのセットに分けられる．Set5 は Zeyde らの手法 [48] で画質評価に使われた画像セットである．Set14 は Bevilacqua らの手法 [49] で画質評価に使われた画像セットである．

計算コストの比較では，GPU などの PC 構成を SRCNN と合わせることが難しいため，処理時間ではなく，積和演算（Multiply and Accumulation; MAC）数を用いて比較を行なった．Glasner らの手法は，入力の低解像度画像から作成したピラミッド画像を用いてパッチの探索を行う手法であり，本手法や SRCNN と比較して処理時間が膨大となる．そのため，積和演算数で計算コストを見積もることが難しいことから，Glasner らの手法の積和演算数の結果は算出していない．また，構築された処理の規模を測るためにパラメータサイズを算出した．SRCNN は，重みの 1 要素を 32bit float として算出した．CGP を用いて構築したグラフ構造式は，ノードの番号（N_{bit}）のビット数を 6bit，演算子（F_{bit}）のビット数を 4bit，出力ノード数を O_{num}，演算ノード数を F_{num}，エッジ数を E_{num} として次の式 6.2 を用いて算出した．

$$\text{Parameter (byte)} = \lceil ((F_{num} + O_{num}) \times N_{bit} + F_{num} \times F_{bit} + E_{num} \times N_{bit})/8 \rceil. \tag{6.2}$$

比較として従来手法から，Interpolation-based の手法である Bicubic，Example-based の手法である Glasner らの手法を用いた．Glasner らの手法は，Yang [45] の Web ページのソースコード[2]を元に筆者らが作成したものを用いた．比較手法のパラメータを表 6.3 に示す．

[1] http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html
[2] http://eng.ucmerced.edu/people/cyang35

表 6.1: CGP の設定

Parameter	Value
Generation alternation model	MGG*
Number of generation	400000
Population size	50
Number of children	20
Crossover rate	1.0
Mutation rate	0.03

* Minimal Generation Gap [5]

表 6.2: CGP で用いた演算子

Function	Number of input	MAC	Description
+	2	1	Add two input
−	2	1	Subtract input2 from input1
Max	2	3	The largest value of inputs
Min	2	3	The smallest value of inputs
Ave	2	4	The average value of inputs
*0.1	1	1	Multiply input by 0.1
*0.25	1	1	Multiply input by 0.25
*0.5	1	1	Multiply input by 0.5
*2.0	1	1	Multiply input by 2.0
*-1.0	1	1	Multiply input by -1.0

表 6.3: 比較手法の設定

Methods	Parameters	Values
Glasner	Patch size	5×5
	Resolution levels	5
	Number of nearest patches	2
	Number of iterations for back projection	3

表 6.4: 各手法の PSNR の評価値と積和演算数

PSNR	Bicubic	SRCNN	Glasner	method1	method2	method3	method1+2	method3+2
Set5	33.66	**36.66**	35.98	34.85	34.98	35.09	35.18	35.13
Set14	30.23	**32.45**	32.09	31.33	31.49	31.46	31.61	31.55
MACs	64	263552		190	9555	5905	7190	26952
Parameters (bytes)	64	32128		312	169	898	413	996

6.3.2 実験結果と考察

PSNR の画質評価と 1 つの画素を 2 倍に拡大するときに必要な積和演算数の結果を表 6.4 に示す．なお，最良の評価値は太字で，2 番目の評価値は下線で表記している．SRCNN が最も PSNR の値が高く，Glasner らの手法が次に高かった．手法 1 + 2 がそれらに次いで 3 番目の結果であった．手法 1，手法 2，手法 3 とそれらを組み合わせた手法である手法 1 + 2，手法 3 + 2 を比較すると手法 1 + 2，手法 3 + 2 共に，組み合わせていない手法より優れていた．このことから，手法 1，手法 2，手法 3 を組み合わせることの有効性を示した．また，本章では手法 1，手法 2 と手法 2，手法 3 をそれぞれ組み合わせたが，人手で各手法の組み合わせを方法を決定するよりも，自動構築することでさらなる精度の向上が期待できる．

SRCNN と比べて，手法 1 + 2 は Set5，Set14 において 1.48dB，0.84dB 低かったが，計算コストは 1/36 倍，パラメータサイズは 1/77 倍であった．

Set5 の PSNR と積和演算数の関係をプロットしたものを図 6.7 に示す．右上にプロットされている手法ほど，画質と計算コストのトレードオフの関係が良好であることを示している．手法 1，手法 2，手法 3 の関係について，手法 1 は，MAC 数が 3 手法の中で最も少なく処理も簡単だが，画質は低い．手法 2 は，パラメータサイズは最も少ないが繰り返し演算を行うため，MAC 数が最も多くなっている．手法 3 は，グラフ構造式の数が多いため MAC 数，パラメータサイズは多くなっているが，テスト画像に様々な画像が含まれているため画質は最も高くなっている．

各手法のテスト画像結果の一部をそれぞれ図 6.3，図 6.4，図 6.5，図 6.6 に示す．図 6.3 において，手法 1 + 2 は目標画像と近く，SRCNN とあまり違いのない画像を出力できている．図 6.4，図 6.5 において，手法 1 + 2 はエッジの周囲にノイズが出ており少し不自然な出力になっているが，目標画像や Glasner，SRCNN に近い画像を出力できている．図 6.6 において，手法 1 + 2 は SRCNN や Glasner と比べて，少しぼけた出力画像となっていた．

これらのことから，提案手法は画質評価値において SRCNN に劣っているが，視覚的に出力画像は SRCNN や目標画像に近い画像を出力できており，少ない積和演算数で処理を行うことができ，パラメータサイズも非常に少ないことから小規模な回路構造で高画質な画像を生成できる手法であるといえる．

PSNR の評価値について，PSNR などの画質評価指標は相対的な評価指標であり，絶対的にこの値ならば十分という値を決めることは難しく，その差も画像によって変動するため視覚的にどの程度の差があるのか判断することは難しい．どの程度の画質，処理速度が必要かはユースケースによってしまう．本手法は，MAC 数，パラメータサイズが少なく高速に処理を行える手法として選択肢の 1 つとなる．

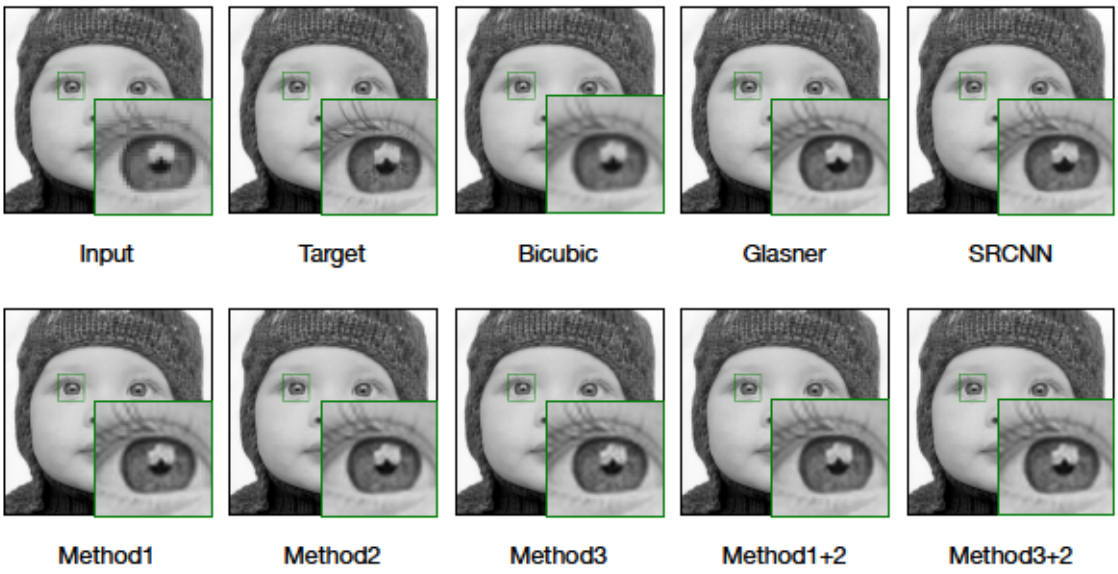

図 6.3: テスト画像結果の一部（Baby from Set5）

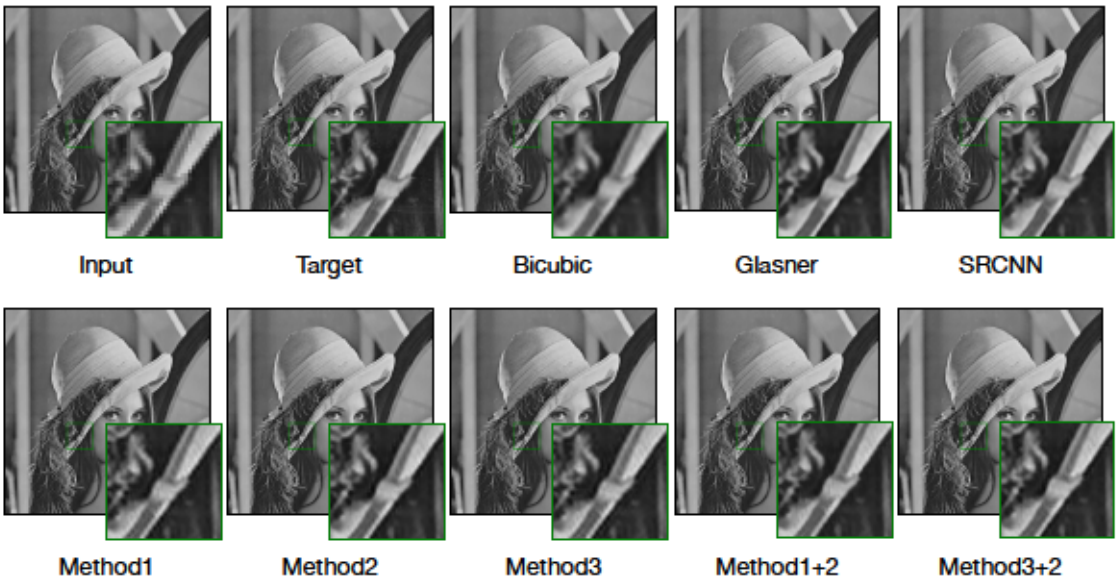

図 6.4: テスト画像結果の一部（Lenna from Set14）

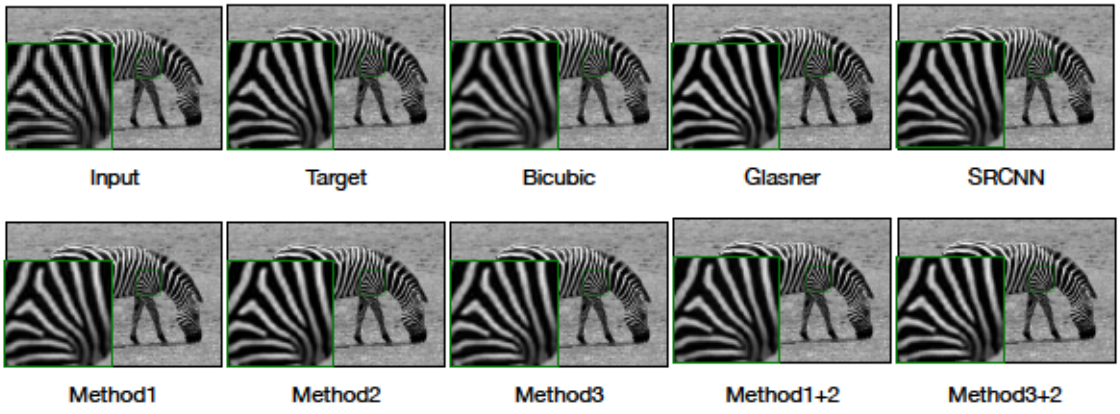

図 6.5: テスト画像結果の一部（Zebra from Set14）

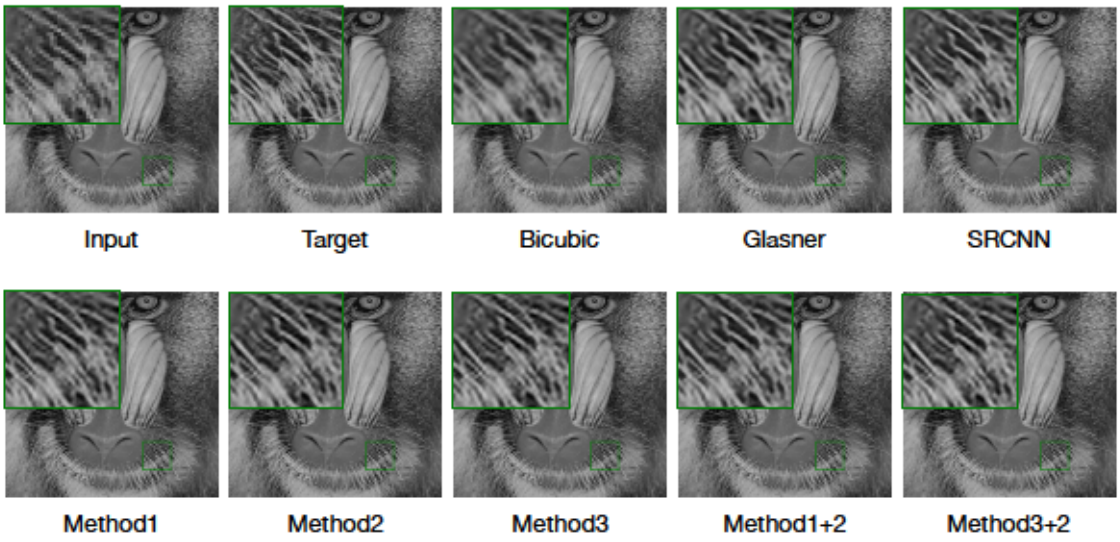

図 6.6: テスト画像結果の一部（Baboon from Set14）

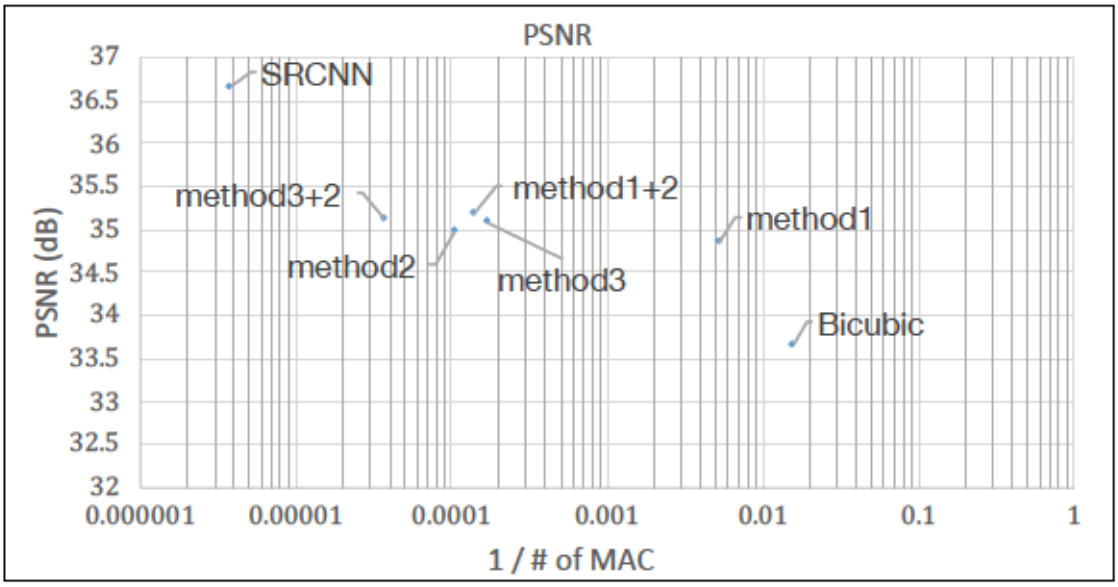

図 6.7: 画質（Set5）と積和演算数の関係

6.4 まとめ

第 3 章，第 4 章，第 5 章で提案した手法を組み合わせた 2 つの手法を提案した．Deep Learning を用いた手法である SRCNN と実験設定を合わせ超解像処理実験を行った．その結果，提案手法は画質評価では SRCNN に劣るが，積和演算数，パラメータサイズが少ない超解像処理を構築できていることを示した．また，これまでの提案手法を組み合わせることで精度が向上することを確認した．

第7章　結論

7.1　本論文で得られた成果

本論文では，局所的な画素情報から高解像度画像の画素値を比較的単純な関数を用いて算出する計算コストの低い処理方式を採用し，グラフ構造式を与えられた学習画像に合わせて進化計算法の1つである Cartesian Genetic Programming（CGP）を用いて設計することで，小規模な回路で，高速かつ高精度な超解像処理を自動構築する手法の提案を行なった．各章で得られた成果は次の通りである．

1. 複数のグラフ構造式をアンサンブルした手法

進化計算法の 1 つである CGP を用いて低解像度画像の 1 つの画素およびその近傍画素の画素値から対応する高解像度画像の画素値を算出するグラフ構造式を自動構築し，シングルフレーム超解像処理を行う手法を提案した．また，複数のグラフ構造式をアンサンブルする超解像処理の構築手法を提案した．傾向の異なる顔画像，建物画像，テキスト画像の３つのカテゴリの画像セットに対して超解像処理実験を行い，複数のグラフ構造式をアンサンブルすることで画質が向上することを示した．従来手法との比較で，提案手法は現状の最高レベルの従来手法に近い画質の画像を高速に生成することができることを示した．

2. 周囲の画素の関係性を考慮した手法

拡大処理によって拡大された画像に対して，周囲の画素の関係性を考慮し繰り返し補正を行う超解像処理の構築手法を提案した．提案手法は，拡大処理と補正処理の 2 つで構成され，それぞれの処理を CGP を用いて構築した．顔画像，建物画像，テキスト画像の傾向の異なる３つのカテゴリの画像セットに対して超解像処理実験を行い，提案手法の有効性を検証した．従来手法との比較で，提案手法は現状の最高レベルの従来手法に近い画質の画像を高速に生成することができることを確認した．

3. 画像の局所的な領域に応じた手法

複数のグラフ構造式を用いたシングルフレーム超解像処理を自動構築する手法を提案した．複数のグラフ構造式を世代数によって切り替えながら最適化することで，それぞれ異なった役割の処理を構築することができることを示した．また，入力画像の領域ごとに適した複数のグラフ構造式を構築することで画質が向上することを確認し，その有効性を示した．提案手法は，最高レベルの手法と比較して近い画質評価値を示しており，高速に超解像処理を行えていることから高速かつ高画質な超解像処理を行える手法であるといえる．

4. 提案手法の組み合わせて DeepLearning を用いた手法との比較

第 3 章，第 4 章，第 5 章で提案した手法を組み合わせた 2 つの手法を提案した．Deep Learning を用いた手法である SRCNN と実験設定を合わせ超解像処理実験を行った．その結果，提案手法は画質評価では SRCNN に劣るが，積和演算数，パラメータサイズが少ない超解像処理

を構築できていることを示した．また，これまでの提案手法を組み合わせることで精度が向上することを確認した．

7.2 今後の課題

本研究における今後の課題を次に示す．

1. **CGP の演算子の検討**
   本研究では，CGP の演算子は計算コストの低い演算を実験的に選択しているが，0.3 倍など表現することが難しい実数倍の演算も存在する．様々な値を選択肢として用意すると探索範囲が広くなり，学習が難しくなる．そのため，できるだけ少ない演算子で処理を構築しているが，演算子のさらなる検討を行うことで，学習を簡単にすることが期待できる．

2. **評価指標の検討**
   本研究では，適応度関数に重み付き絶対値誤差，SSIM を用いたが，画質評価指標に用いていた PSNR は２乗誤差から算出される．そのため，絶対値誤差は２乗誤差と相関が高いが，２乗誤差を適応度関数に用いることで精度の向上を期待できる．また，重み画像として Sobel フィルタのエッジ強度を用いているが，構築済みの処理の出力画像と目標画像から重みを算出することが考えられる．

3. **適切な学習画像の選択**
   第６章では，エッジ強度で重み付けを行い学習画像のサンプリングを行ったが，画像の局所的なパターンの出現性や低解像度と高解像度画像の差の大きさなどを考慮した適切な学習画像を選択することで超解像処理の性能向上を期待できる．また，学習の経過に応じてパターンの多さや難しさを変えた学習画像を切り替えることが考えられる．

4. **拡大処理の数の自動決定**
   第３章，第５章では，グラフ構造式の数を実験的に３，５と決定している．また，第６章では，第３章，第４章，第５章の手法を組み合わせることで精度を向上することを確認した．人手で手法の組み合わせを方法を決定するよりも，本研究で得られた知見を元に自動構築することで，グラフ構造式の適切な数や組み合わせを探索することができ，さらなる精度の向上が期待できる．

謝辞

博士課程後期進学の機会をいただき，本研究を進めるにあたり親身なご指導ご助言，豊かな研究環境を賜りました長尾智晴先生に深く感謝いたします．

本論文をまとめるにあたり貴重なご指導，ご助言をいただきました田村直良先生，森辰則先生，岡嶋克典先生，富井尚志先生，白川真一先生に感謝いたします．

そして，長尾研究室の皆様には大変お世話になりました．特に，先に卒業された先輩や同期の皆様からはご指導していただいたり貴重なご意見をいただきました．感謝いたします．

また，博士課程進学を認めてくださった上長の方々，博士号取得を応援していただき貴重なご意見をくださりました [ET 開] の皆様，いつも昼食をご一緒させていただいている皆様，以前同じ職場で働いていた皆様に感謝いたします．

最後に，博士号取得を温かく見守ってくれた家族や友人の皆様に感謝いたします．この中の 1 人でも欠けていたら本論文を執筆することはできませんでした．本当に，ありがとうございました．

参考文献

[1] Holland John. Holland, adaptation in natural and artificial systems, 1992.

[2] David Edward Goldberg. Genetic algorithm in search. *Optimization and Machine Learning*, 1989.

[3] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, Vol. 1. MIT press, 1992.

[4] Julian F. Miller and Peter Thomson. Cartesian genetic programming. *Processing of the Third European Conference on Genetic Programming (EuropeGP2000).*, Vol. 1802, pp. 121–132, 2000.

[5] 佐藤浩, 小野功, 小林重信. 遺伝的アルゴリズムにおける世代交代モデルの提案と評価. 人工知能学会誌, Vol. 12, No. 5, pp. 734–744, 1997.

[6] Lukáš Sekanina. Image filter design with evolvable hardware. In *Applications of Evolutionary Computing*, pp. 255–266. Springer, 2002.

[7] Tomáš Martínek and Lukáš Sekanina. An evolvable image filter: Experimental evaluation of a complete hardware implementation in fpga. In *Evolvable Systems: From Biology to Hardware*, pp. 76–85. Springer, 2005.

[8] Simon Harding and Wolfgang Banzhaf. Genetic programming on gpus for image processing. *International Journal of High Performance Systems Architecture*, Vol. 1, No. 4, pp. 231–240, 2008.

[9] Simon Harding, Vincent Graziano, Jürgen Leitner, and Jürgen Schmidhuber. Mt-cgp: Mixed type cartesian genetic programming. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pp. 751–758. ACM, 2012.

[10] Gnu image manipulation program （gimp）. `www.gimp.org`.

[11] Michal Irani and Shmuel Peleg. Improving resolution by image registration. *CVGIP: Graphical models and image processing*, Vol. 53, No. 3, pp. 231–239, 1991.

[12] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, Vol. 20, No. 3, pp. 21–36, 2003.

[13] J.D. Van Ouwerkerk. Image super-resolution survey. *Image and Vision Computing*, Vol. 24, No. 10, pp. 1039–1052, 2006.

[14] Xin Li and Michael T. Orchard. New edge-directed interpolation. *Image Processing, IEEE Transactions on,*, Vol. 10, pp. 1521–1527, 2001.

[15] S Battiato, G Gallo, and F Stanco. A locally adaptive zooming algorithm for digital images. *Image and Vision Computing*, Vol. 20, No. 11, pp. 805 – 812, 2002.

[16] Sebastiano Battiato, Giovanni Gallo, and Filippo Stanco. Smart interpolation by anisotropic diffusion. In *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on*, pp. 572–577. IEEE, 2003.

[17] Claude E. Duchon. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology*, Vol. 18, No. 8, pp. 1016–1022, 1979.

[18] Michal Irani and Shmuel Peleg. Motion analysis for image enhancement: Resolution, occlusion, and transparency. *Journal of Visual Communication and Image Representation*, Vol. 4, No. 4, pp. 324–335, 1993.

[19] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Image super-resolution using gradient profile prior. In *Computer Vision and Pattern Recognition(CVPR), 2008. IEEE Conference on*, pp. 1–8. IEEE, 2008.

[20] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Gradient profile prior and its applications in image super-resolution and enhancement. *Image Processing, IEEE Transactions on*, Vol. 20, No. 6, pp. 1529–1542, 2011.

[21] Yu-Wing Tai, Shuaicheng Liu, Michael S Brown, and Stephen Lin. Super resolution using edge prior and single image detail synthesis. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2400–2407. IEEE, 2010.

[22] W.T. Freeman, Egon C Pasztor, and Owen T Carmichael. Learning low-level vision. *International journal of computer vision*, Vol. 40, No. 1, pp. 25–47, 2000.

[23] W.T. Freeman, T.R. Jones, and E.C. Pasztor. Example-based super-resolution. *Computer Graphics and Applications, IEEE*, Vol. 22, No. 2, pp. 56–65, 2002.

[24] Jian Sun, Nan-Ning Zheng, Hai Tao, and Heung-Yeung Shum. Image hallucination with primal sketch priors. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, Vol. 2, pp. 729–736. IEEE, 2003.

[25] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. In *Computer Vision and Pattern Recognition(CVPR), 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, Vol. 1, pp. I–I, June 2004.

[26] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *Computer Vision and Pattern Recognition(CVPR), 2008. IEEE Conference on*, pp. 1–8. IEEE, 2008.

[27] Jianchao Yang, J. Wright, T.S. Huang, and Yi Ma. Image super-resolution via sparse representation. *Image Processing, IEEE Transactions on*, Vol. 19, No. 11, pp. 2861–2873, Nov 2010.

[28] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 349–356, 2009.

[29] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5197–5206. IEEE, 2015.

[30] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *arXiv preprint arXiv:1501.00092*, 2014.

[31] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 38, No. 2, pp. 295–307, 2016.

[32] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1646–1654, 2016.

[33] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 370–378, 2015.

[34] Ding Liu, Zhaowen Wang, Bihan Wen, Jianchao Yang, Wei Han, and Thomas S Huang. Robust single image super-resolution via deep networks with sparse prior. *IEEE Transactions on Image Processing*, Vol. 25, No. 7, pp. 3194–3207, 2016.

[35] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision*, pp. 391–407. Springer, 2016.

[36] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1637–1645, 2016.

[37] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. *arXiv preprint arXiv:1704.03915*, 2017.

[38] 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.

[39] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[40] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *arXiv:1710.01992*, 2017.

[41] 永田智洋, 長尾智晴. セルラ進化型ニューラルネットワークによるシングルフレーム超解像処理. 電子情報通信学会論文誌, Vol.J95-D, No.10, pp. 1859–1868, 2012.

[42] Nvidia cuda: Compute unified device architecture. `http://developer.nvidia.com/object/cuda.html`.

[43] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[44] Caltech buildings dataset.

[45] Chih-Yuan Yang, Jia-Bin Huang, and Ming-Hsuan Yang. Exploiting self-similarities for single frame super-resolution. In *Computer Vision–ACCV 2010*, pp. 497–510. Springer, 2011.

[46] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, Vol. 13, No. 4, pp. 600–612, 2004.

[47] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

[48] Roman Zeyde, Michael Elad, and Matan Protter. *On Single Image Scale-Up Using Sparse-Representations*, pp. 711–730. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[49] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012.

研究業績リスト

論文誌

夏井裕介，長尾智晴，Cartesian Genetic Programming を用いたシングルフレーム超解像処理の自動構築. 電子情報通信学会論文誌 D Vol.J97-D, No.11, pp.1641-1650, Nov. 2014.

国際会議発表

Yusuke Natsui and Tomoharu Nagao, Automatic Construction of Single Frame Super-Resolution Using Cartesian Genetic Programming, Proc. of The 6th International Workshop on Computational Intelligence & Applications 2013 (IWCIA 2013), pp.149-154, July 13, Hiroshima City Univ., Hiroshima, Japan (2013)

Yusuke Natsui and Tomoharu Nagao, Single Frame Super-Resolution Using Multiple Graph Structured Program, Proc. of The 9th International Workshop on Computational Intelligence & Applications 2016 (IWCIA 2016)

国内学会発表

夏井裕介，長尾智晴：進化計算法を用いたシングルフレーム超解像における補間ルールの獲得，情報処理学会第 75 回全国大会，2013 年 3 月 6 日〜8 日，東北大学，宮城 (2013)

夏井裕介，長尾智晴：進化計算法を用いたシングルフレーム超解像処理の自動構築，情報処理学会第 76 回全国大会，2014 年 3 月 12 日，東京電機大学，東京 (2014)

夏井裕介，長尾智晴：学習の効率を考慮したシングルフレーム超解像処理の自動構築， 2014 年度画像電子学会第 42 回年次大会， 2014 年 6 月 29 日，早稲田大学，東京 (2014)

夏井裕介，長尾智晴 複数のグラフ構造プログラムに基づくシングルフレーム超解像処理の自動構築，情報処理学会第 77 回全国大会，5ZG-04，2015 年 3 月 19 日，京都大学，京都 (2015)

受賞

情報処理学会第 75 回全国大会　学生奨励賞

情報処理学会第 76 回全国大会　学生奨励賞

解説論文

夏井裕介, 長尾智晴. Cartesian Genetic Programming を用いたシングルフレーム超解像処理の自動構築, 「画像ラボ」日本工業出版株式会社, Vol.27, No.7, pp 20-25, 2016.

付録A　獲得されたグラフ構造

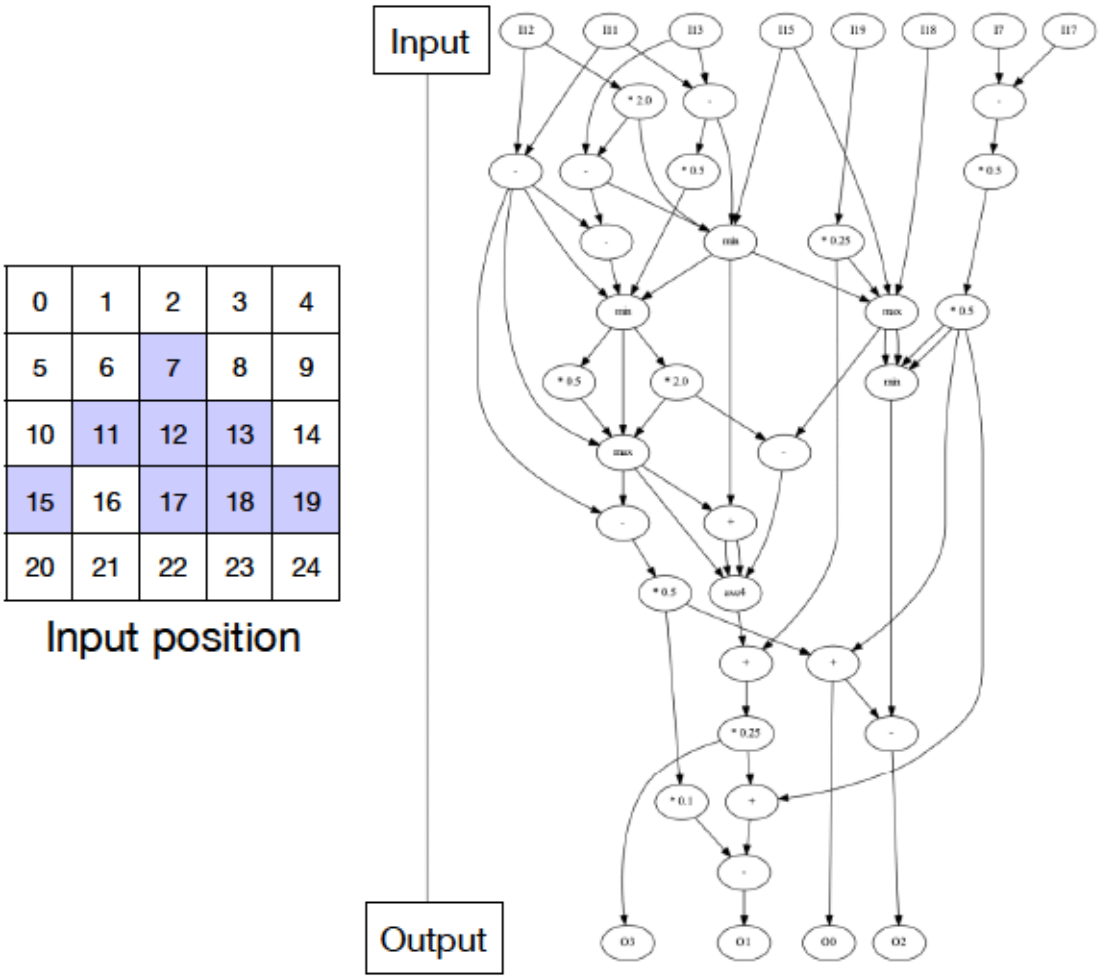

図 A.1: 3 章の実験において獲得された CGP_{enl} の構造（Building）

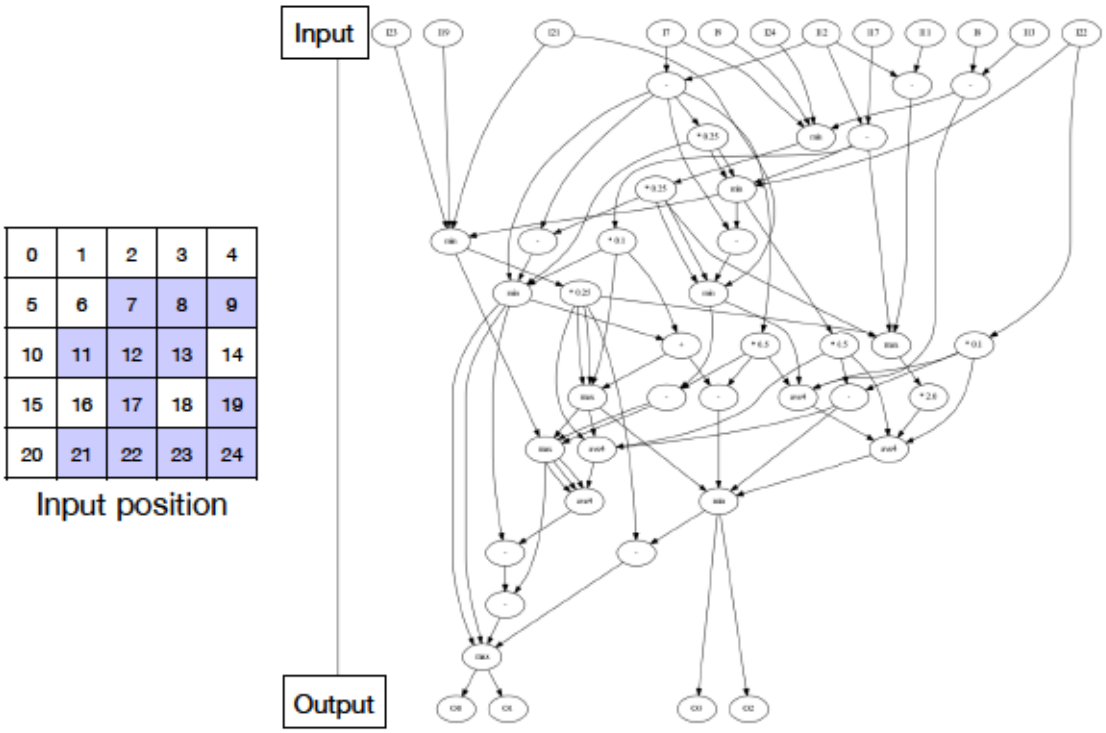

図 A.2: 3 章の実験において獲得された $CGP_{up}1$ の構造（Building）

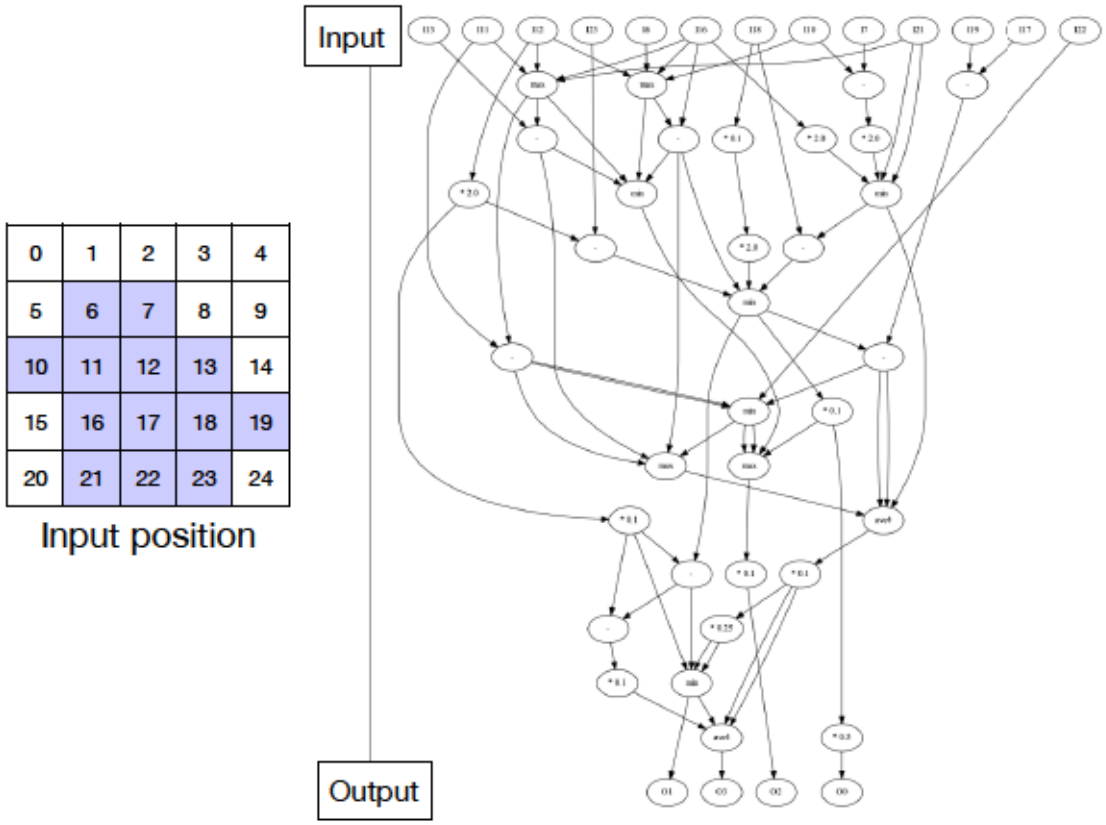

図 A.3: 3 章の実験において獲得された $CGP_{up}2$ の構造（Building）

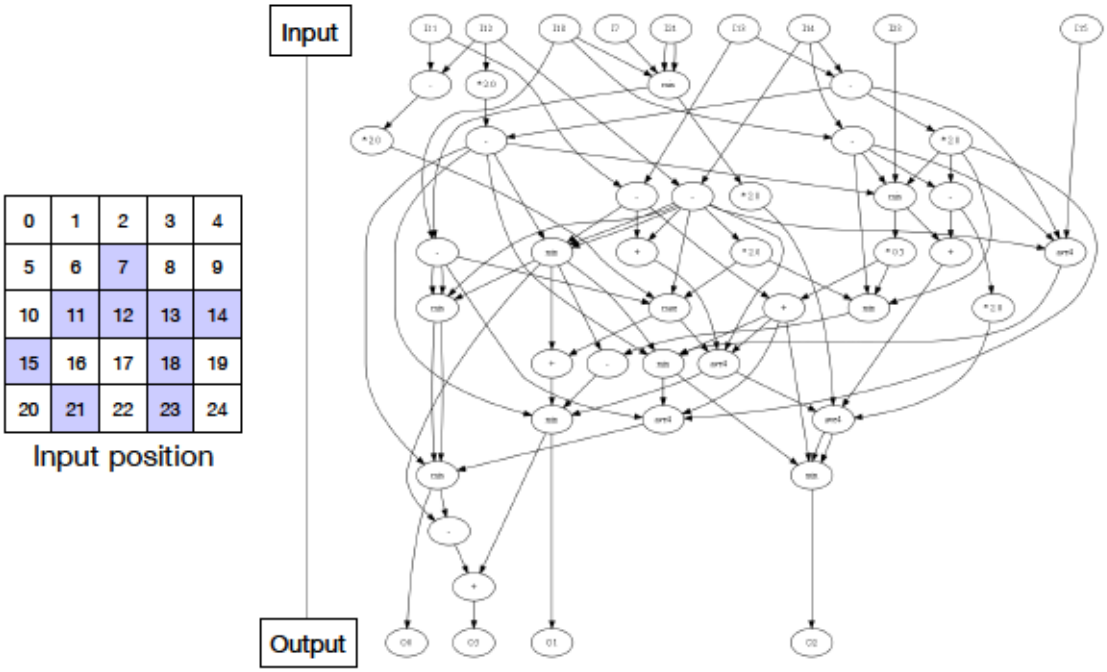

図 A.4: 3 章の実験において獲得された CGP_{enl} の構造（Text）

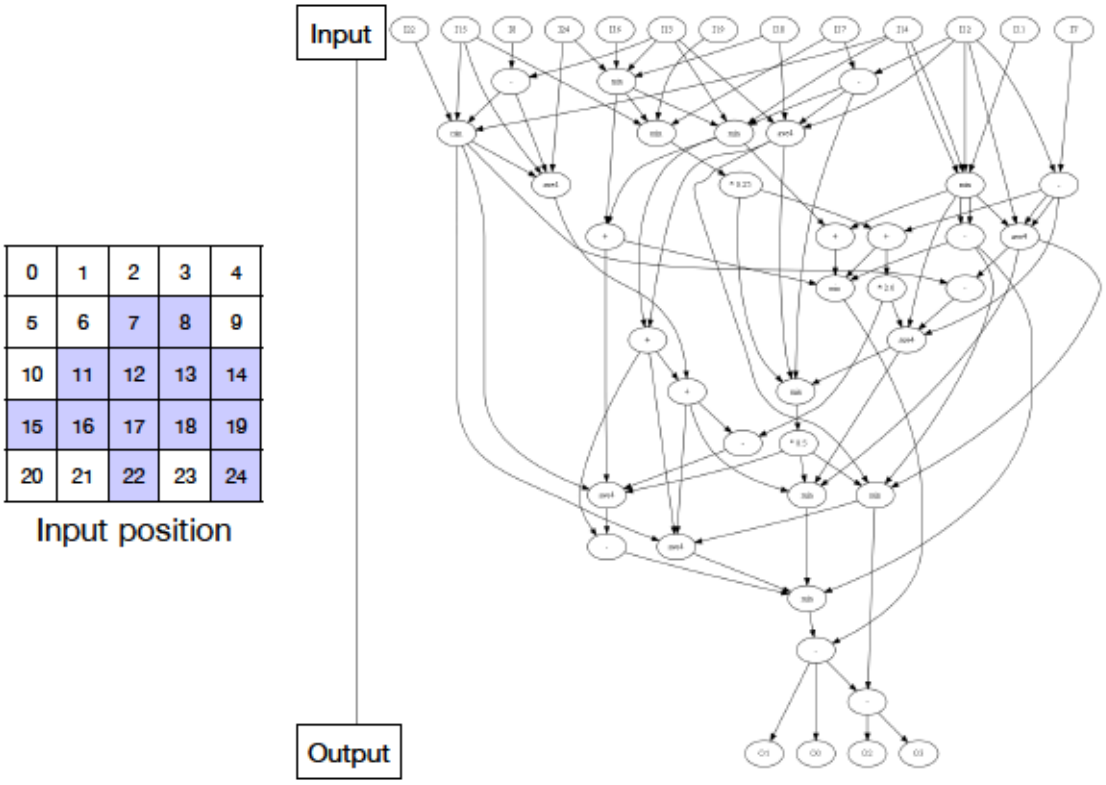

図 A.5: 3 章の実験において獲得された $CGP_{up}1$ の構造（Text）

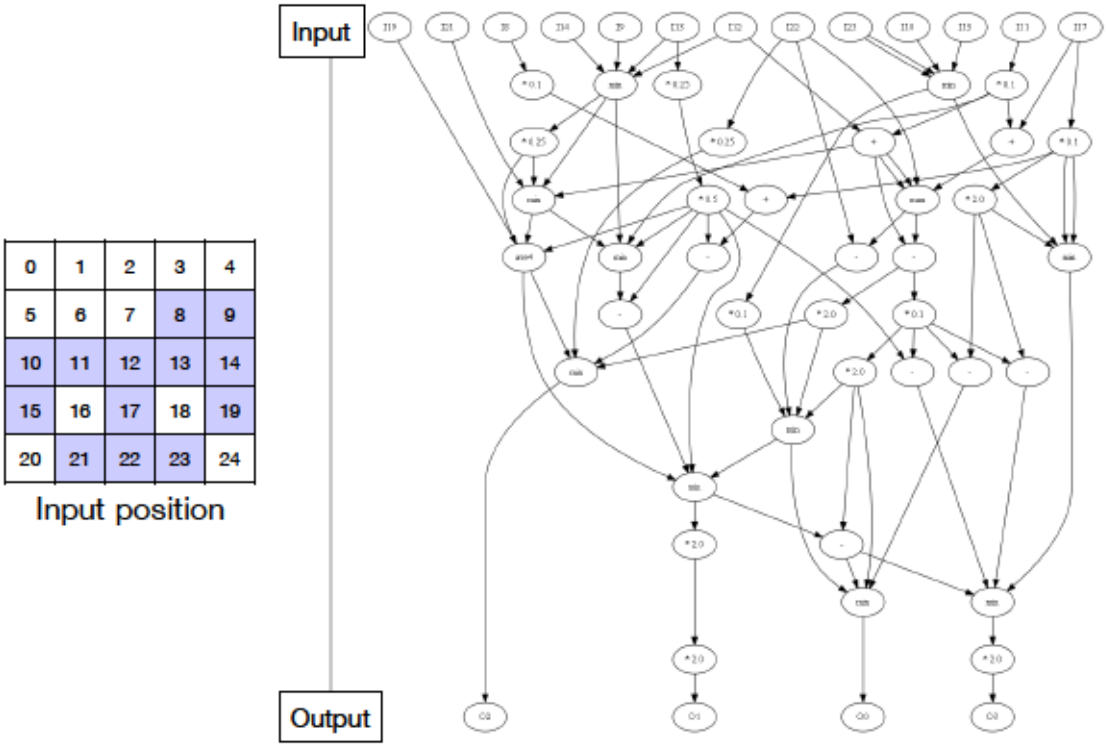

図 A.6: 3 章の実験において獲得された $CGP_{up}2$ の構造（Text）

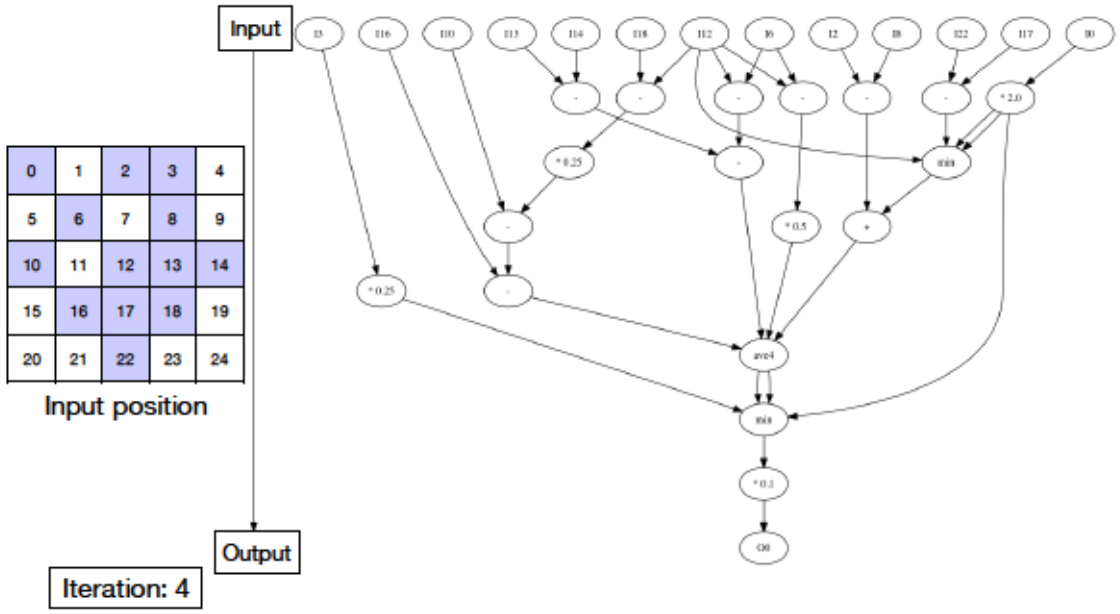

図 A.7: 4 章の実験において獲得された $CGP_{mod}1$ の構造（Face）

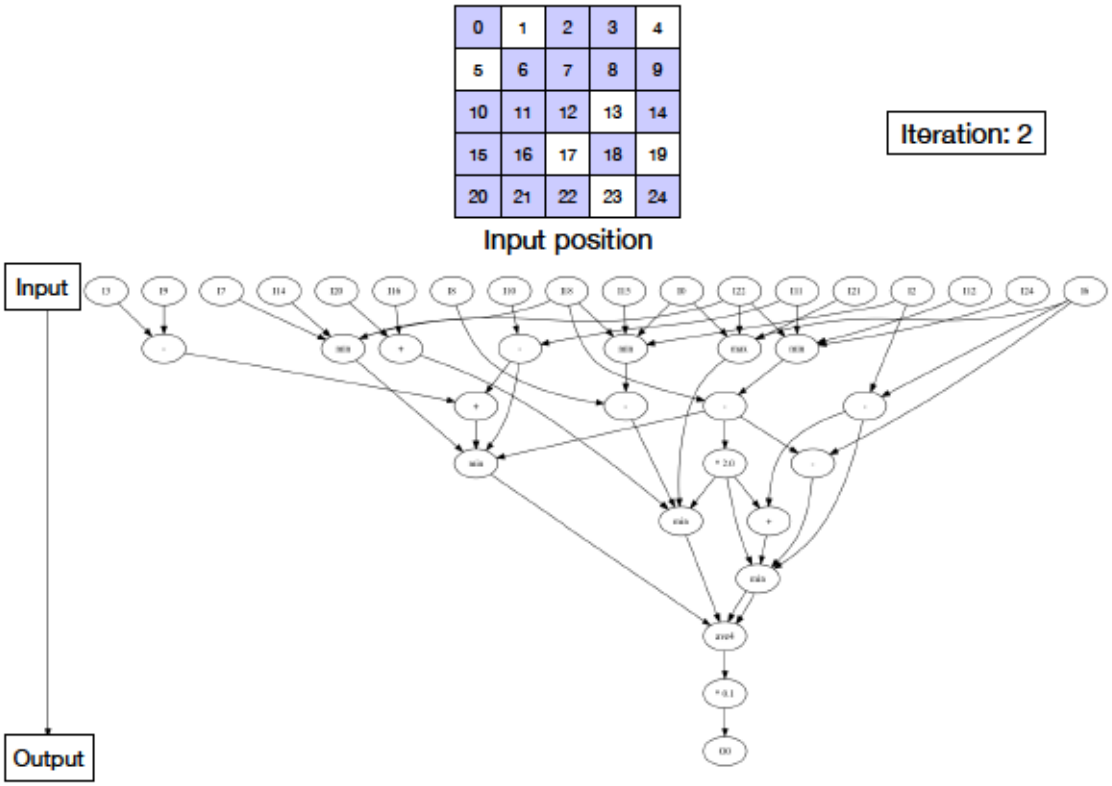

図 A.8: 4 章の実験において獲得された $CGP_{mod}1$ の構造（Building）

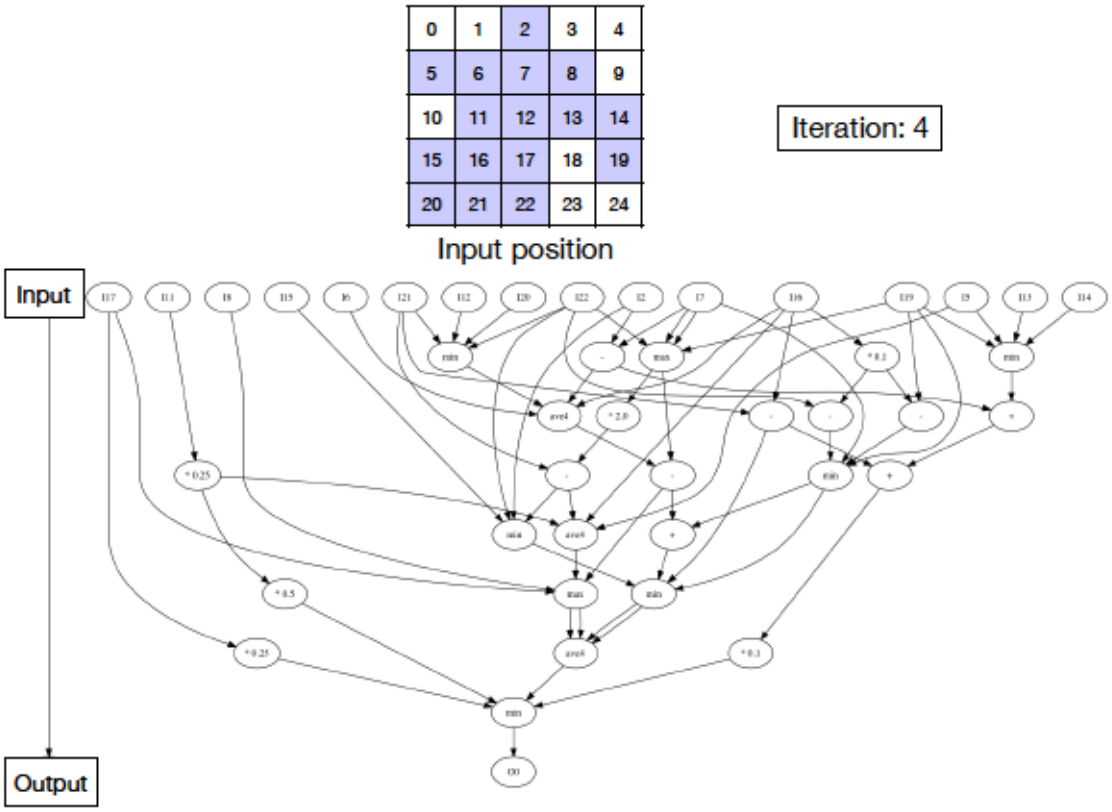

図 A.9: 4 章の実験において獲得された $CGP_{mod}1$ の構造（Text）

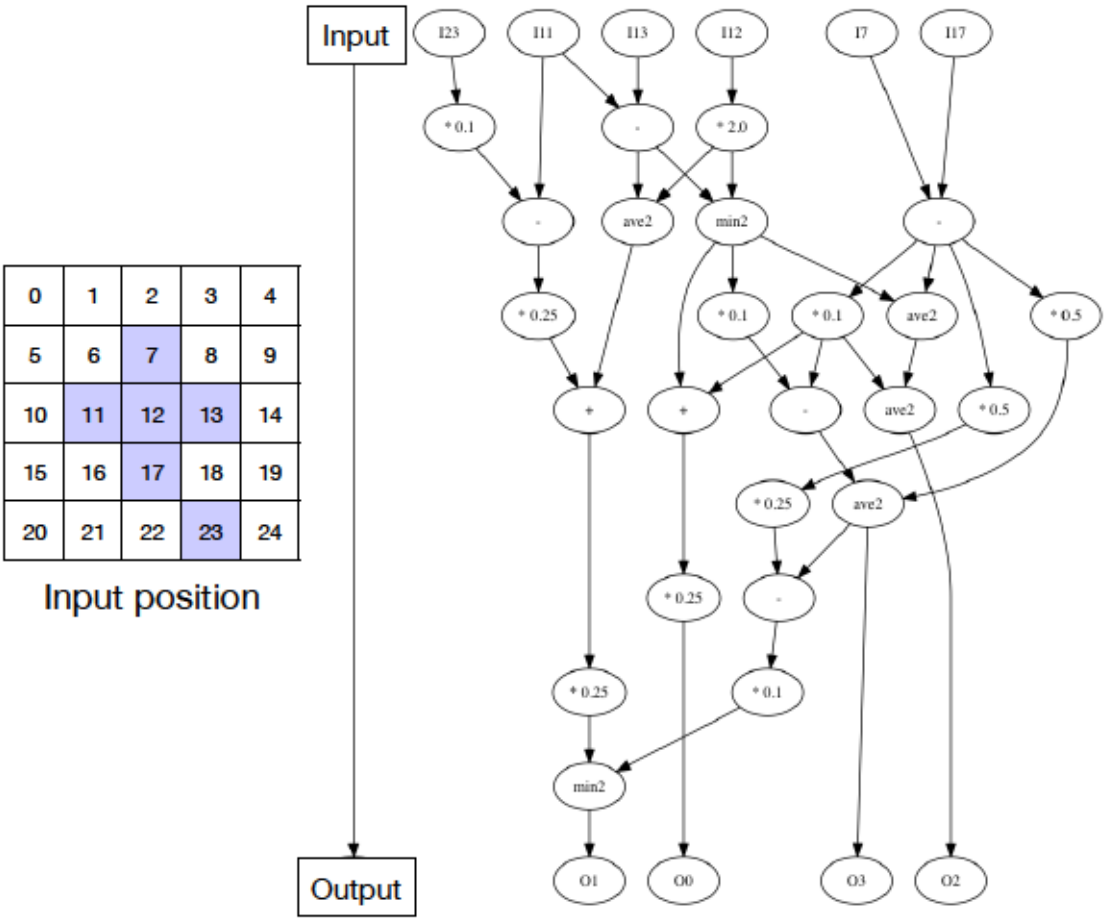

図 A.10: 5 章の実験において獲得された CGP_{enl} の構造

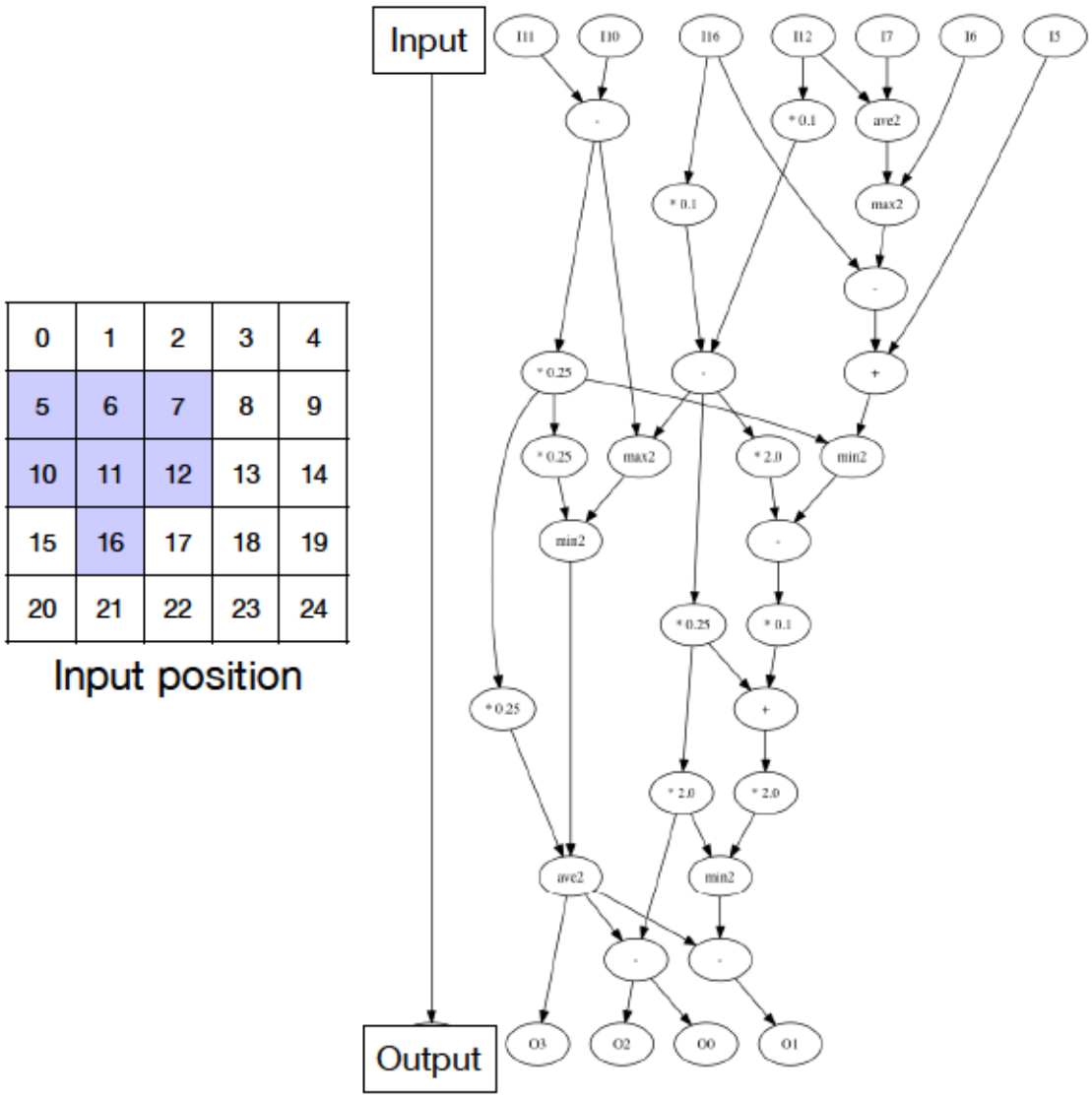

図 A.11: 5 章の実験において獲得された CGP_{up1} の構造

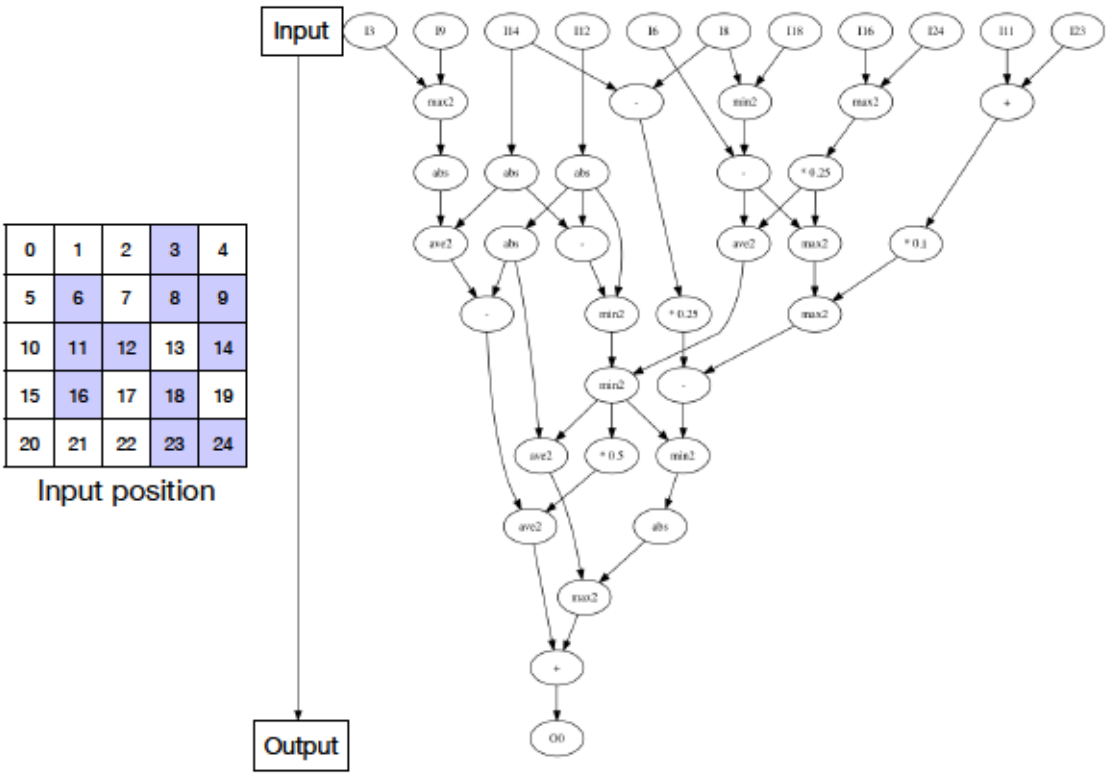

図 A.12: 5 章の実験において獲得された CGP_{conv1} の構造

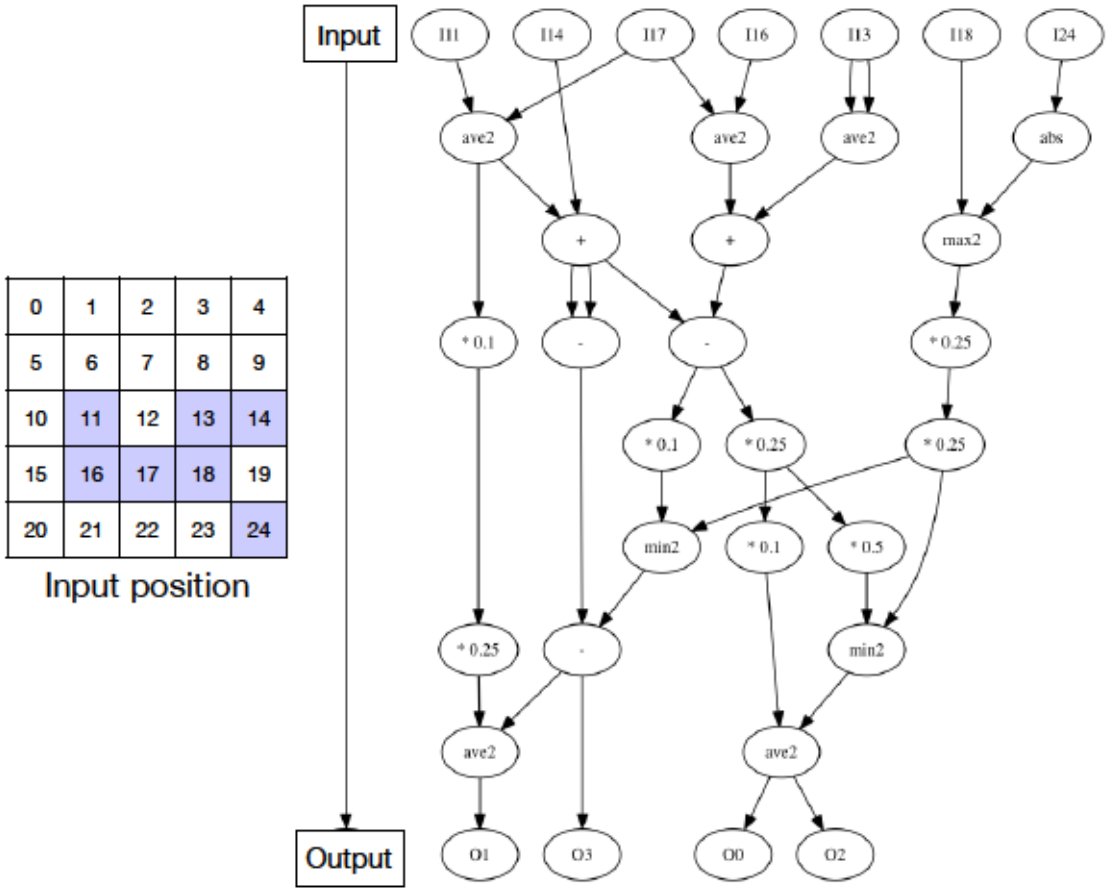

図 A.13: 5 章の実験において獲得された CGP_{up2} の構造

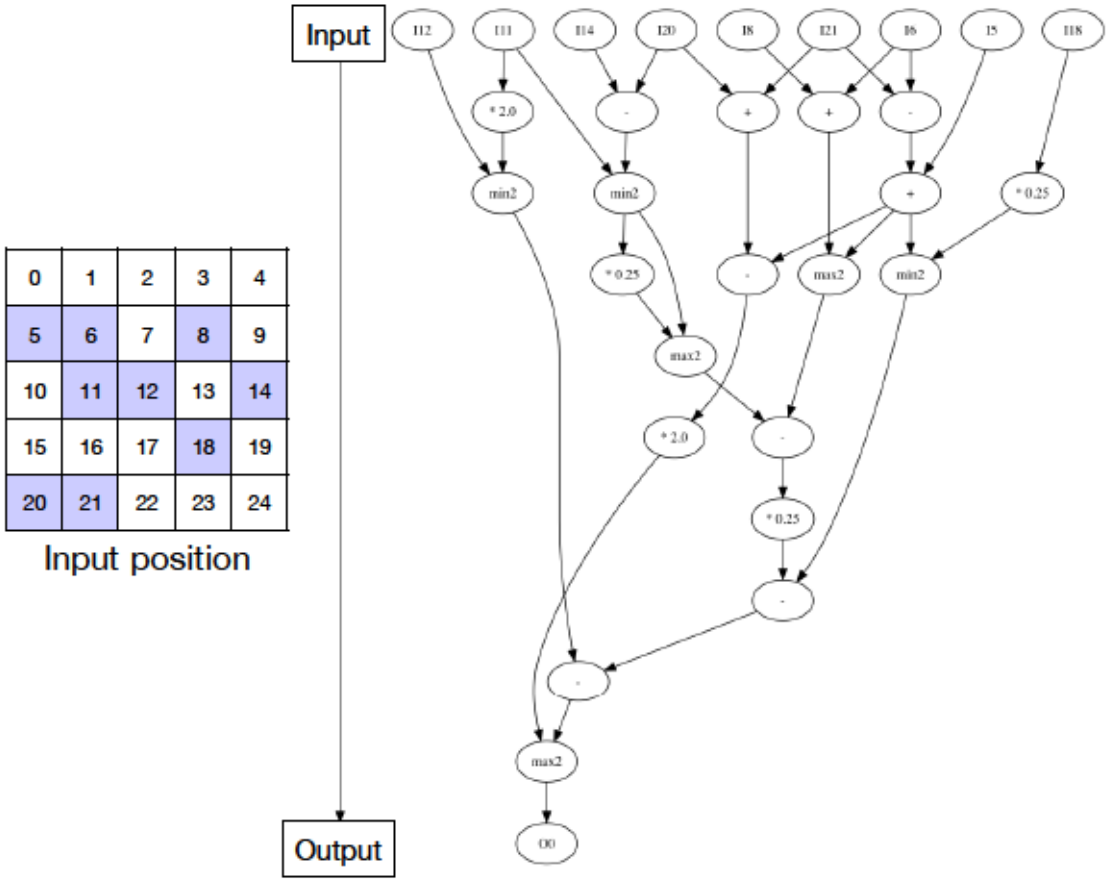

図 A.14: 5 章の実験において獲得された CGP_{conv2} の構造

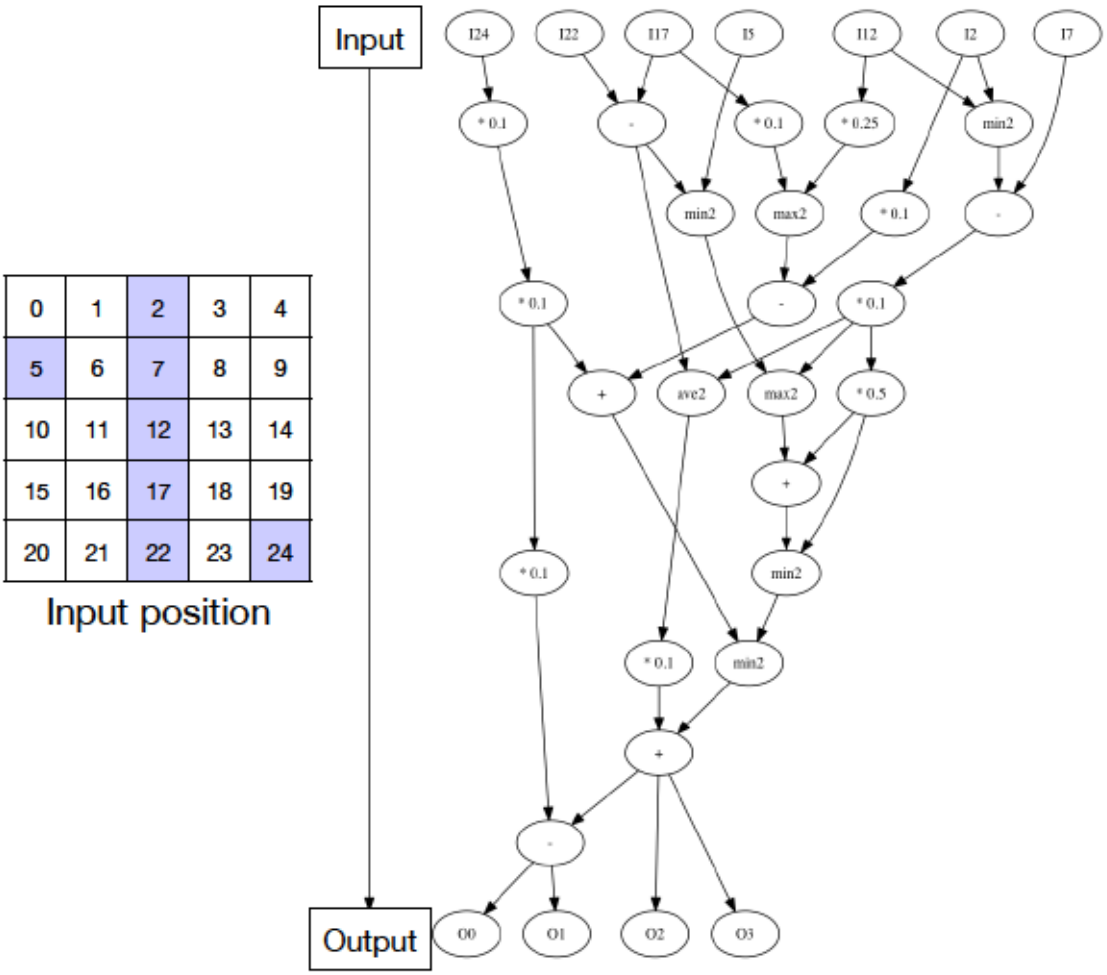

図 A.15: 5 章の実験において獲得された CGP_{up3} の構造

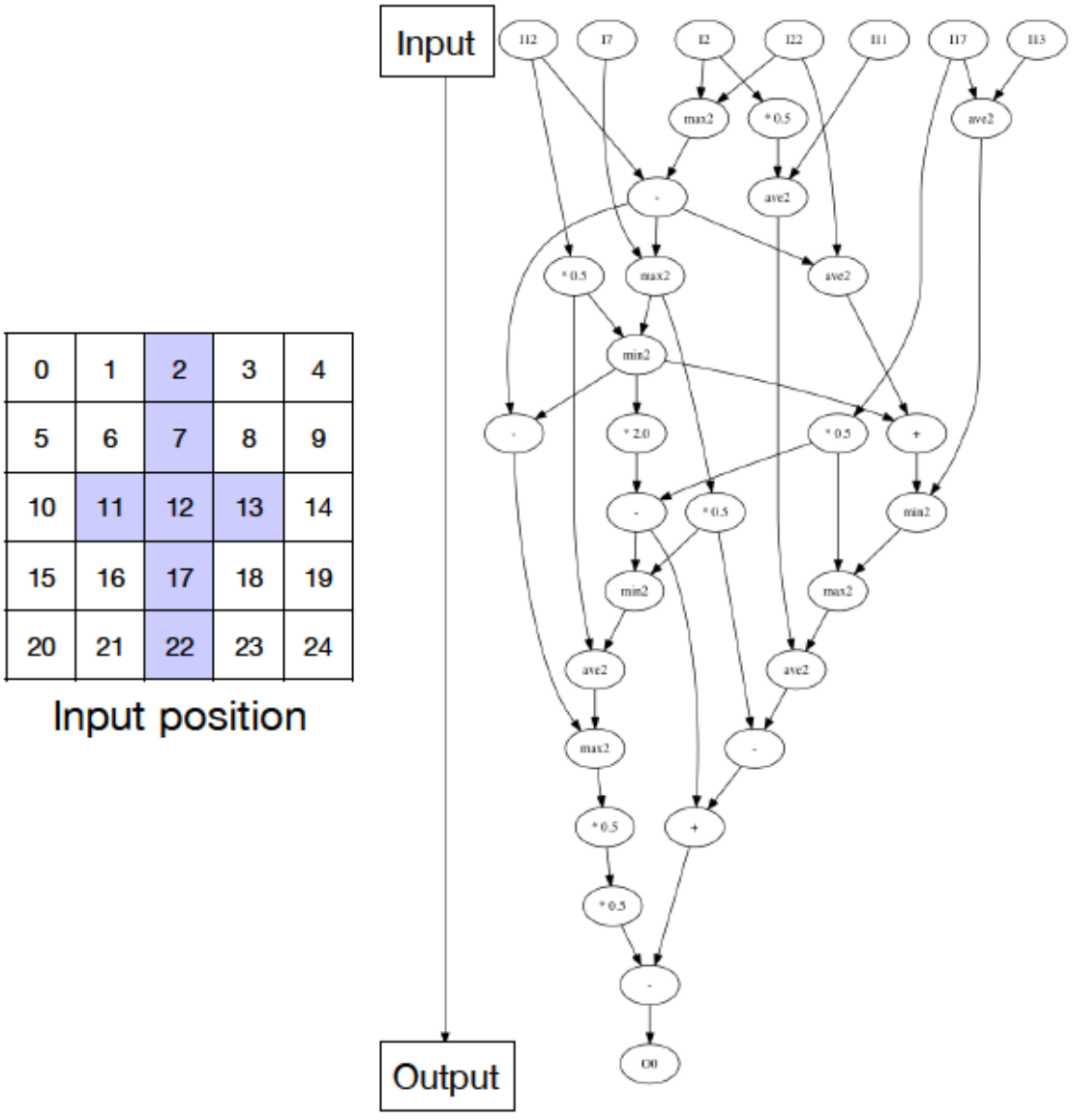

図 A.16: 5 章の実験において獲得された CGP_{conv3} の構造

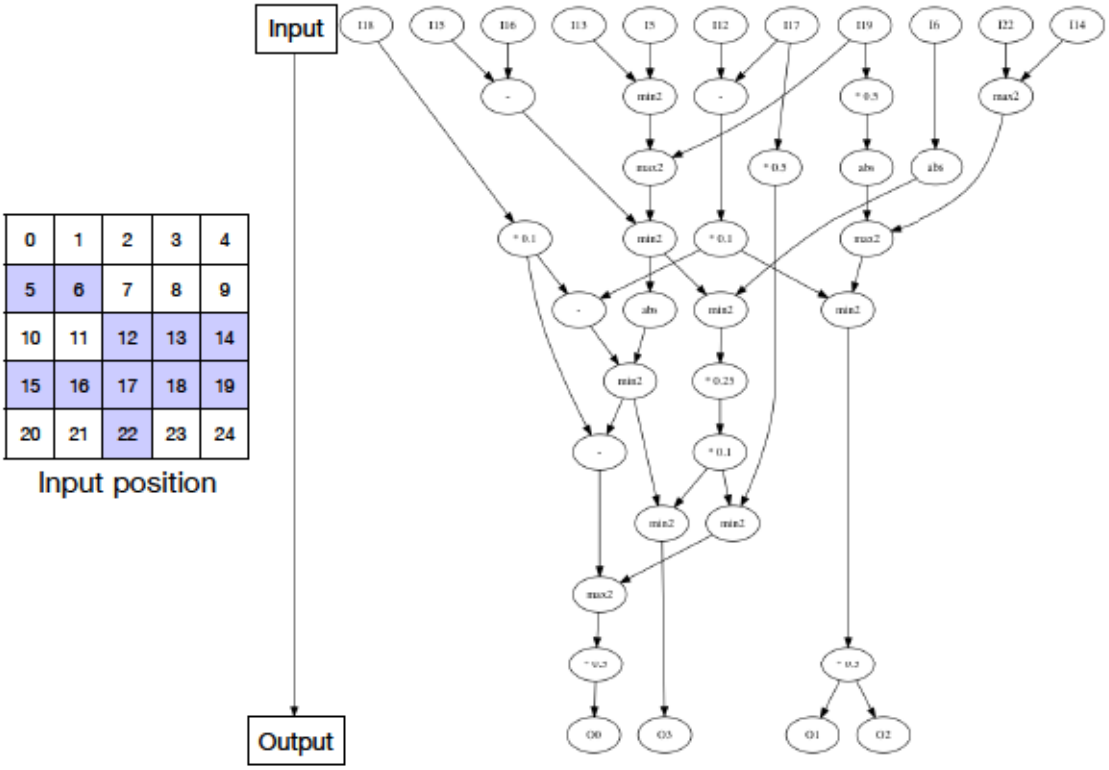

図 A.17: 5 章の実験において獲得された $CGP_{up}4$ の構造

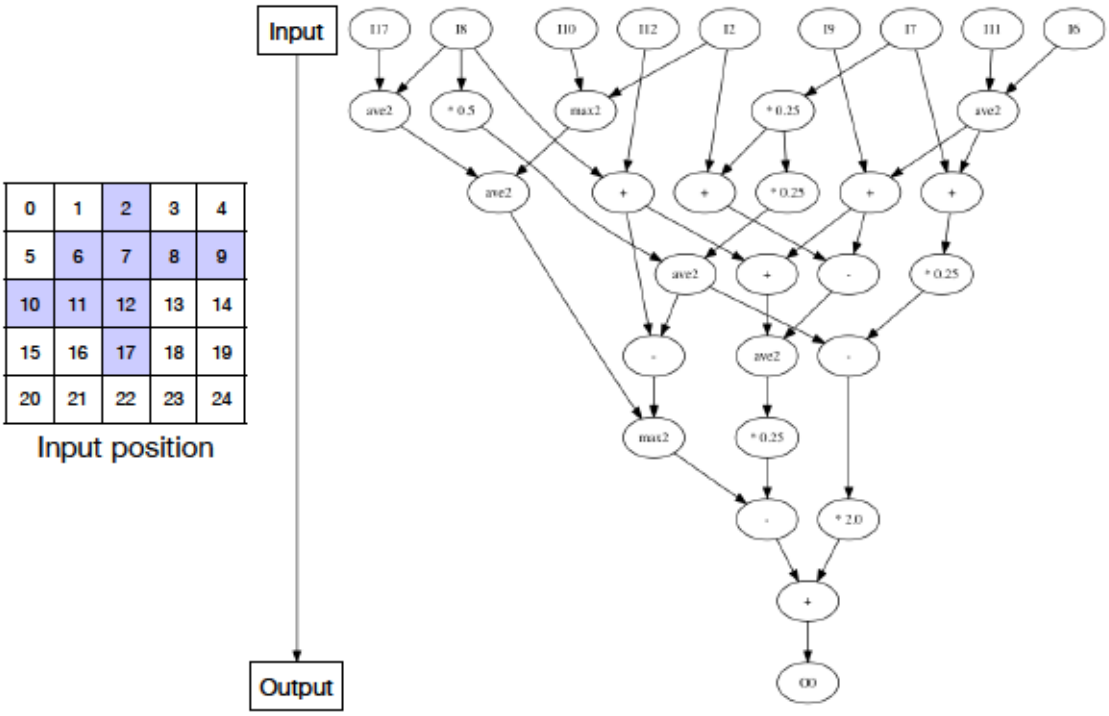

図 A.18: 5 章の実験において獲得された $CGP_{conv}4$ の構造

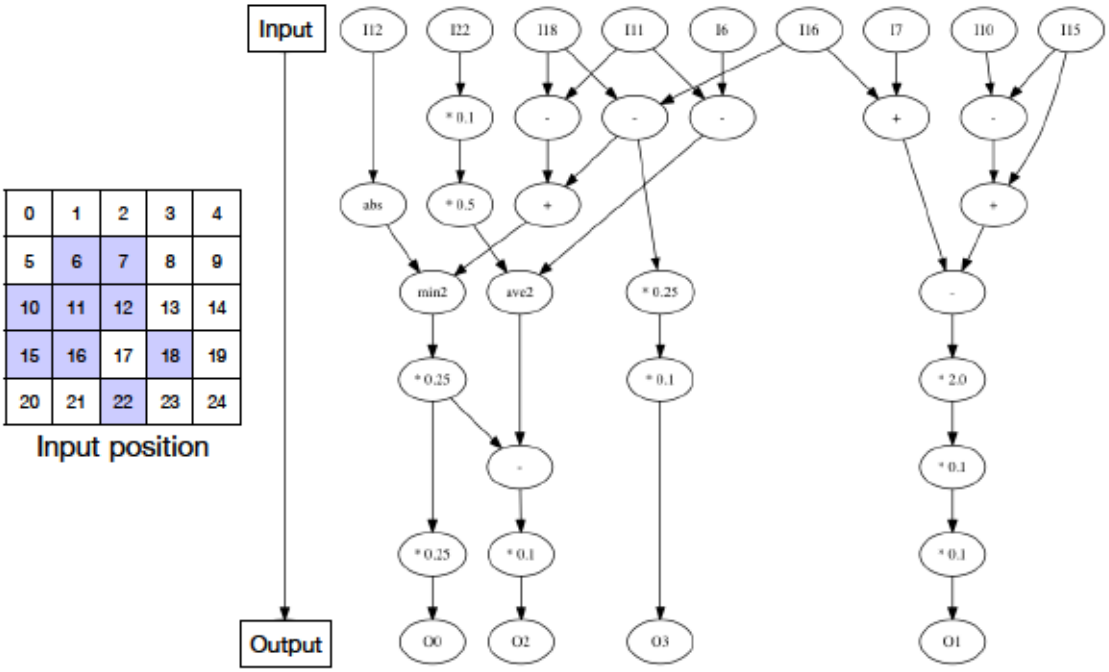

図 A.19: 5 章の実験において獲得された $CGP_{up}5$ の構造

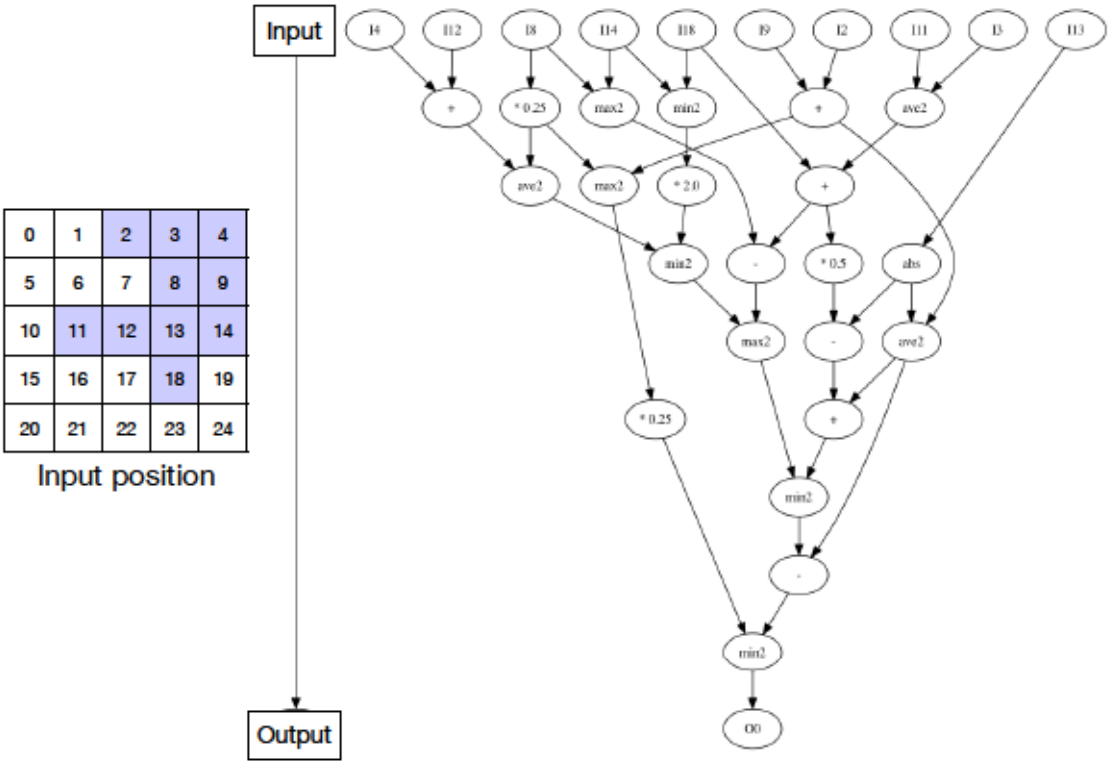

図 A.20: 5 章の実験において獲得された $CGP_{conv}5$ の構造