Vol.2020-CVIM-220 No.19
2020/1/24

# 深層学習を用いた医療画像補間

山下滉生[†1]
会津大学院大学[†3]

## 概要

この研究の目的は、ディープラーニングを使用して低品質の医療画像を高品質の画像に変換する画像強化システムを構築することである。そのために、私たちは超解像技術を私たちの環境に合わせて改善し実験を行った。

私は導入として、Convolutional Auto Encoder(CAE)と Super Resolution Convolutional Neural Network(SRCNN)[1]の 2 種類のシステムを実装した。この研究の医療画像としては、福島県立医科大学からの光コヒーレンストモグラフィ（OCT）スキャンデータの目の画像を入力データおよびターゲットデータとして使用します。

## 1. はじめに

身の回りの生活において、写真は人々にとってとても必要不可欠なものになっている。また、写真とはその場の風景や実際のものを残すものなのだ。その写真は拡大すると解像度が低い画像に限っては詳細に見ることができない場合が存在する。それは医療画像も例外ではない。

福島医科大学では、目の OCT スキャンを撮ったとき、その画像は低画質であり詳細にデータを見ることができない。そのため、画像を多く撮って、それらの画像の加算結果を出力することで、より詳細にみることがわかった。しかし、その多くの画像を撮ることが莫大な時間を使用してしまう。だから、莫大な時間を使用しなくても一枚の画像から高画質なもの(加算結果)を出力できるようにしたいと考えた。

その場合に Super-resolutiont というものが、とても役に立つのである。Super Resolution は解像度をあげる技術である。この研究はその Super-resolution の中でも Deep Learning を使った Super-resolution を使用し、この課題を解決していきたいと考えている。私は SR システムのようなものを医用画像でも同じことができると考えて適用した。実験した出力結果および学習器の内容について説明する。

## 2. メソッド

### 2.1 畳み込み演算

畳み込み演算は畳み込み層で行う処理のことを言い、画像処理でのフィルター演算に相当する。この演算方式は、画像関連の AI を開発するに当たってとても重要な演算方法であり、ニューラルネットワークが発展した理由の一つである。この演算処理は大まかな流れとしては、入力画像の左上の端っこから指定したフィルタの大きさを取得しその画像データ（X）を以下の計算式（４）にフィルタ（K）と一緒に代入する。

$$f(x) = \sum_{i}^{k} x_i k_i + b \tag{1}$$

計算結果を一つの出力のピクセルの値とし出力の対応する場所に格納していく。指定した間隔(ストライド)で、スライドさせながら同じことを繰り返し右下端まで繰り返す。具体的に Figure1 を使って説明する。

Figure1 では、黄色く塗ってある部分があるがそれは、’zero padding’いった手法で入力データを拡張している。それは、入力データの周り全てに 0 を代入し入力データのサイズを大きくするといった手法で、出力が適切な大きさになるように調整している。この padding を何行何列入れるのかによって出力のサイズを調整するのである。このサイズは、$O_h$,$O_w$を出力のサイズ(h:高さ、w:幅)、フィルタのサイズを$F_h$,$F_w$、パディングのサイズを P、ストライドを S として置き、以下の(2)(3)[2]の式に代入して計算をして適切な値を考える。

$$O_h = \frac{H + 2P - F_h}{S} + 1 \tag{2}$$

$$O_w = \frac{H + 2P - F_w}{S} + 1 \tag{3}$$

Figure1 の場合、フィルタが 3x3 であるため、入力データの左上端から 3x3 分だけ抜き出し(赤枠)、計算式（1）に抜き出した入力データ（X）とフィルタ（K）を代入する。式に代入した結果 2 が出力され出力データの一番初めである左上のピクセルに代入される。そのあと、指定したストライドで左にスライドし同じことを繰り返すのである。一行目の右端までいったらストライドのたての方向にもスライドし、また右端から同じことを繰り返す。 最終的に右下までこれを繰り返すのである。Figure1 の場合、フィルタが 3x3 であるため、入力データの左上端から 3x3 の大きさの分だけ入力データを抜き出す(赤枠)。そして、計算式（1）に抜き出した入力データ（X）とフィルタ（K）を代入する。式に代入した結果 2 が出力され出力データの一番初めである左上のピクセルに代入される。そのあと、指定したストライドで左にスライドし同じことを繰り返すのである。一行目の右端までいったらストライドのたての方向にもスライドし、また右端から同じことを繰り返す。最終的に右下までこれを繰り返し行う。

深層学習を用いた医療画像補間
†1 KOKI YAMASHITA, University of Aizu

ⓒ 2020 Information Processing Society of Japan

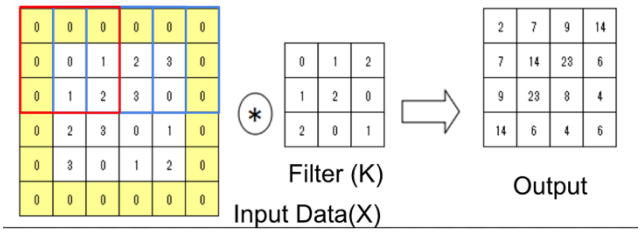


Figure 1: 畳み込み演算の処理方法

### 2.2 Convolutional Auto Encoder

CAE とは CNN による Auto Encoder(AE)である。一枚の画像に対して、2.2 の Convolutional Operation を何層にも分けて畳み込みを行う。Fig2[3]では 28x28x1 のイメージに対して、14x14x32 になるように Stride＝２として畳み込みを行う。それを三回行い、最後に Flatten という平滑化を行い潜在表現を取り出す。そのあと全結合層（FC）に入れ、特徴部分が取り出された画像データを一つのノードに結合し、活性化関数によって変換された値（特徴変数）を出力する。最後に Deconv は前半の畳み込み層で使った重みとバイアスと足りないノードは outpadding という削除される偽の出力を割り当てることと見なしたノードを使用し、画像をたたみこんだ分だけ再構築を行う。これが基本的な CAE の流れだ。

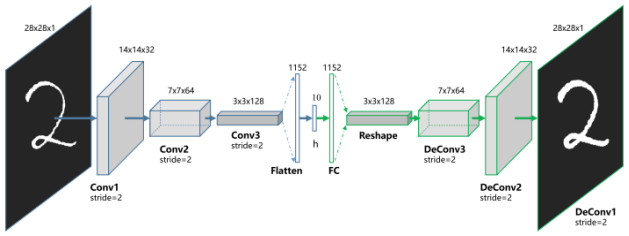


Figure 2: Convolutional Auto Encoder の簡単な構造

### 2.3 Super Resolution Convolutional Neural Network

このモデルは、低解像度画像と高解像度画像間のエンドツーエンドのマッピングを直接学習し、最適化以外の前処理/後処理はほとんどいらないといった物であったため、当時は、それ以前の主流であったスパースコーディングベースの SR メソッドより容易に学習でき、精度も良いといったモデルでした。操作としては、まずバイキュービック補間をして目的のサイズにアップスケールを行うのですが、本研究のデータセットの特徴である目的変数と入力変数が同じ大きさだから必要ないので使用しない。CNN の以下のような三層でできている。

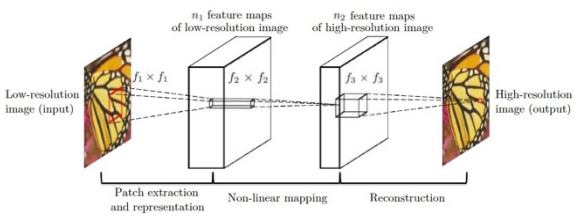


Figure 3: SRCNN の全体構造

1) Patch extraction and representation(filter:9x9)
2) Non-linear mapping (filter:1x1)
3) Reconstruction(filter:5x5)

この三層でできている。

1)では、LRimage からパッチを重複して抽出を行なっている。言い換えれば、「小領域を切り出す」といった物である。2)でパッチごとに解像度に変更してる。3)最後にパッチごとに行った高解像度パッチを集約して、高解像度の画像を表現している。学習時の損失関数として以下の平均二乗和誤差(4)を私たちは採用をしている。

$$MSE = \frac{1}{\mathrm{mn}} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - O(i,j)]^2 \qquad (4)$$

以上のことが SRCNN の概要である。。

### 2.4 検証指標

この研究では、PSNR インデックスと SSIM インデックスを使用して評価してる。 PSNR は、(4)の ＭＳＥ と下記の式（5）で定義されている。 m と n は画像の垂直および水平サイズであり、MAX_I は画像の可能な値の最大ピクセル値である。 また、入力イメージを I として、出力イメージを O として定義する。

$$PSNR = 10 * log_{10}\left(\frac{MAX_I^2}{MSE}\right) \qquad (5)$$

PSNR は、Peak signal-to-noise ratio の略である。 これは、信号が受け取ることができる最大電力と劣化の原因となるノイズの比率を表す工学用語であり、画質の再現性に影響を与えている。 端的にいうのであれば、画像がどの程度劣化したかを示す値を表現している。値が小さいほど劣化が大きくなり、値が大きいほど元の画像に近くなる。 この調査では、出力層アクティベーション関数がシグモイドに設定されているため、MAX_I は 1 である。 また PSNR が画像の類似性の指標である理由も説明する。 MSE が大きいほど 2 つの画像は異なるため、PSNR は[4]から小さくなります。したがって、ＰＳＮＲ値が小さいほど画像が劣化している（異なる）と言える。[4]

また、SSIM インデックスは、次の 2 つの式（6）および（7）で定義されている。$x$ および $y$ は、モデルの出力画像および目的変数である。 $\mu$ は平均、σ は分散を意味している。 ただし、$\sigma_{xy}$は $x$ と $y$ の共分散を意味している。

$$SSIM_i(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \qquad (6)$$

$$c_1 = (k_1 L)^2, c_2 = (k_2 L)^2 \qquad (7)$$

SSIM は、Structural similarity の略語である。 式（7）で、2 つの変数が弱い分母で除算を安定させるという

$c$ を説明する。 また、L はピクセル値のダイナミックレンジを意味している。 この調査では、学習データではこのデータがフロート型であるため、L は 1.0 に設定されている。 また、$k_1$と$k_2$はデフォルトの数値である 0.01 と 0.03 を使用している。 SSIM は、2 つの画像間の類似性を測定するために使用されている。 言い換えれば、SSIM インデックスは完全な参照メトリックである。つまり、画質の測定または予測は、参照としての初期の非圧縮または歪みのない画像に基づいている。 [5]

## 3. データベース

ここでは私たちが用いるデータセットのことを説明する。

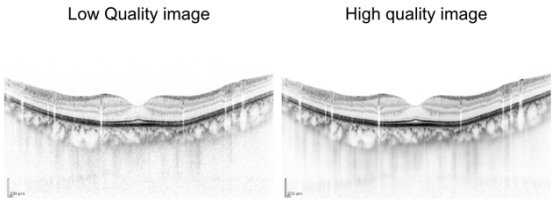


Figure 4:データベースのサンプルペアデータ

Figure5 では、今回使用したデータのサンプルを添付します。私たちはこのような pair の画像群を 11pair 所持しています。一枚当たりの大きさが(678,847)くらいのグレースケール画像で、その 11 組のうち、8 組をトレーニングに使用し、3 組がテスト用に振り分けた。その中からトレーニングデータを 3:1 の割合(Train:6pairs val:2 pair)で Train と Val で分けている。元のデータが少ないため、Training データを増やすのに以下二つの工程を行った。各学習者を入力するために、元の入力データと元のターゲットデータの両方から同じ場所を切り取るクロッピング方法である。各画像からランダムに 227x227 の大きさで２０枚ずつクロッピングを行なっている。そのあとに 4 段階の affine 変換(左右反転,上下反転,左右 90 度回転)をそれぞれを保存し、Training データとして保存した。しかし、バリデーションデータでも同じことを行ったがクロッピング作業は行っていない。理由としては、バリデーションデータはテストデータと同じ大きさではならないといけなければならないのとそのデータとして２枚の画像だけでは少なかったからである。次の Table 1 で。Train と Val、Test をそれぞれの画像数を表として表している。

| | Train | Val | Test |
|---|---|---|---|
| 画像の数 | 600 | 10 | 3 |

Table 1: データベースの画像数

## 4. 実験方法

私たちはフレームワークとして pytorch [6]で実装し実験を行なった。 まず最初に必要な処理はデータの正規化である。平均を 0 に、標準偏差を 1 に設定します。これにより、最大値と最小値の差がなくなります。 その後、3 の Table1 で記載したように分割を行なった。そして、学習器を学習させる際に、大きさ自体は関係ないため、画像一枚一枚から左上から patch size を 41 とし、Stride を 14 としてパッチ処理を行いモデルの訓練を行いました。バリデーションとテストデータは何も処理を行わずそのままの画像を入力した。最適化関数として多くの関数を利用したが下記に示しているのは Adam を使用している。この最適化関数が一番よかったからである。活性化関数として平均二乗和誤差を使用している。

## 5. 結果

それぞれの学習器の結果を表示する。
CAE の結果は以下の通りである。

| 学習率 | PSNR | SSIM |
|---|---|---|
| 0.0002 | 28.827 | 0.661 |
| 0.0003 | 28.222 | 0.623 |
| 0.0004 | 28.205 | 0.623 |
| 0.0005 | 27.941 | 0.601 |
| 0.0006 | 27.124 | 0.616 |

Table 2: CAE の結果

SRCNN の結果は以下の通りである。

| 学習率 | PSNR | SSIM |
|---|---|---|
| 0.0002 | 35.054 | 0.942 |
| 0.0003 | 34.702 | 0.941 |
| 0.0004 | 34.772 | 0.942 |
| 0.0005 | 34.626 | 0.941 |
| 0.0006 | 34.796 | 0.942 |

Table 3:SRCNN の結果

二つの表から全体的に SRCNN の画像の方が PSNR も SSIM も高いことがわかります。そして Figure5,6 に二つの画像はの学習器のサンプルの出力を示す。

ⓒ 2020 Information Processing Society of Japan

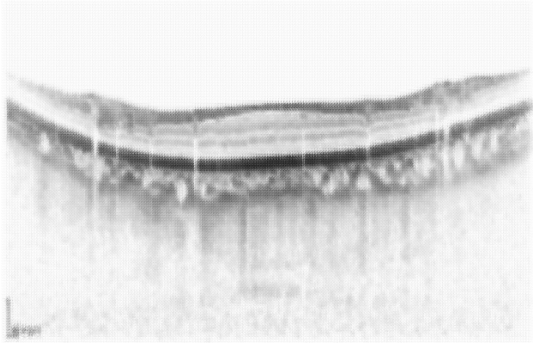

Figure 5:CAE の出力画像

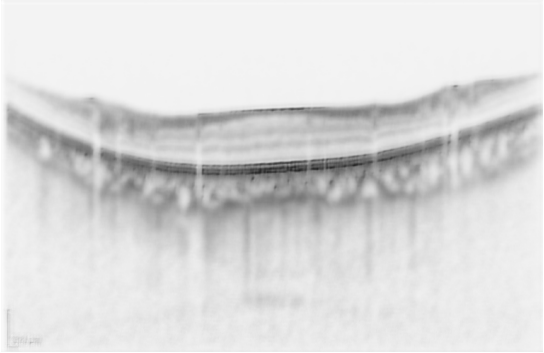

Figure 6: SRCNN の出力画像

## 6. まとめ

SRCNN の方が CAE より知覚的にも良い結果となったが、これは、CAE だと中間層で特徴量を減らしているためだと予測ができる。CAE は深い層を試そうとすると入力のサイズが制限されていくのでやはり、畳み込み層で圧縮していくのは現実的ではない。だから、SRCNN をベースとしたまだ試行していない優秀な SR メソッドがあるのでそれらを今後は研究に生かしていきたいと考えている。

## 参考文献

[1] C. Dong, C.Change Loy Member, ”Image Super-Resolution Using Deep Convolutional Networks”, IEEE,vol. 31,Jul.2015.

[2] K. Saito; ”Deep Learning making from zero” , O’Reilly Japan, Inc. 2016, pp.55-67

[3] Xifeng Guo, Xinwang Liu, En Zhu, Jianping Yin. “Deep Clustering with Convolutional Autoencoders”. ICONIP 2017. [online] https://link.springer.com/chapter/10.1007/978-3-319-70096-0_39

[4] A. Eden; “No-Reference Estimation of the Coding PSNR for H.264-Coded Sequences” https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4266957

[5] Z. Wang; E.P .Simoncelli,; A.C. Bovik, “Multiscale structural similarity for image quality assessment”, Asilomar Conference on Signals, Systems and Computers, 2004. 2. pp. 1398–1402 Vol.2(2003-11-01).

[6] Pytorch document version 0.4.1[online] https://pytorch.org/

ⓒ 2020 Information Processing Society of Japan

# 正誤表

下記の箇所に誤りがございました．お詫びして訂正いたします．

| 訂正箇所 | 誤 | 正 |
|---|---|---|
| タイトル | 深層学習を用いた医療画像補間 | 深層学習を用いた医療画像補正 |
| | | |

ⓒ2020 Information Processing Society of Japan