複雑な劣化画像を学習させたVision Transformerによる単一画像超解像

M2022SC002 林　亮佑
指導教員：河野　浩之

1　超解像の必要性

現在，衛星技術との進歩により様々なアプリケーションで衛星画像が使用されている．実際に衛星画像の需要は増加すると予想されており表 1 によると衛星画像の市場規模は 2030 年には 2022 年の約 4.3 倍の 141.8 億ドルまで増加する．需要の増加の背景として災害時の地上の状態の把握や防衛・安全保障といった緊急事態の対応で衛星画像の利用が増加していることがある．

しかし，衛星画像のような遠距離から撮影された画像は拡大などをする必要があり，拡大過程で劣化する場合がある．劣化した画像を用いて物体認識を行うと認識率が低下してしまう恐れがある．そのため超解像を用いることで劣化を復元し，認識率を向上させることが可能となる．

表 1　衛星画像の市場規模 [1]

年	市場規模
2022 年	32.7 億ドル
2020 年	141.8 億ドル

本研究では現実世界の劣化画像に対応した超解像モデルを提案する．学習に用いる劣化画像はぼかし，リサイズ，ノイズ，JPEG 圧縮を組み合わせることで複雑な劣化表現を獲得する．そして，損失関数に知覚損失関数を用いることでより知覚的な品質が高い超解像画像を得ることを目的とする．

2　超解像手法の先行研究

超解像手法の先行研究を表 2 に示す．先行研究の各モデルの特徴として，CNN ベースの real-ESRGAN は Residual in Residual Dense Block という CNN の層と層の間を残差接続で密につなげることで CNN の深いネットワークを構築している．

次に CNN と Transformer を組み合わせた ACT では CNN の局所的な特徴量を抽出する能力と Transformer の大局的な特徴量を抽出する能力を掛け合わせることでより高い超解像性能を発揮している．

そして Transformer ベースの SwinIR は Vision Transformer のアテンション機構の計算をより効率良く行う Shift-Window 方式を用いた Swin Transformer を用いることで Transfomer のメリットである低パラメータ数に加え，計算効率を向上させている．

本研究では Swin Transformer に複雑な劣化画像を学習させることで CNN ベースの手法よりもパラメータを削減し，高い超解像性能を発揮させることを狙う．

表 2　超解像手法の先行研究

モデル名	特徴
real ESRGAN[2]	blur, noise, resize, JPEG を用いた劣化画像で学習
ACT[3]	CNN と ViT を合わせたモデルを提案 局所的・非局所的な特徴を抽出
SwinIR[4]	SwinViT を超解像に採用 Swin 機構により計算速度を向上

3　劣化画像を学習させた ViT の提案手法

3.1 では使用する超解像モデル，3.2 では Shift Window MSA，3.3 では劣化画像の生成フロー，3.4 ではぼかし付加手順，3.5 ではリサイズ手順，3.6 ではノイズ付加手順，3.7 では JPEG 圧縮手順について説明する．

3.1　使用する超解像モデル

本研究では低パラメータにおいても高い超解像性能を発揮した Swin Transformer をベースに超解像モデルを構築する．

Swin Transformer の構成は LayerNorm の後に shift window multi-head self-attention を行い LayerNorm と MLP へ通す一連の流れを Swin Transformer Layer(STL) としてこの STL を 6 つの後に 1x1 の畳み込みを行う Swin Transformer Block(STB) を使用する．

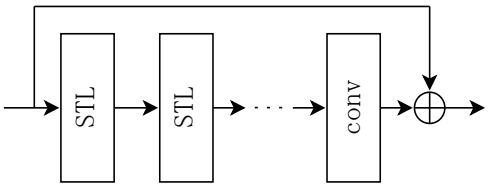


図 1　Swin Transformer Block(STB)

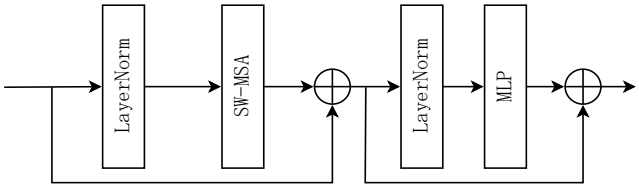


図 2　Swin Transformer Layer(STL)

3.2　Shift Window MSA

Transformer では画像をパッチごとに切り分けて Multi Self Attention(MSA) を行うが，通常 MSA では特徴マッ

プが $h \times w$ 個のパッチがあるすると計算量は $4hwC^2 + 2(hw)^2C$ となる．一方，Swin Transformer ではまず Window Attention によって切り分けられたパッチを Window サイズ M ごとで計算するので計算量は $4hwC^2+2M^2hwC$ となる．よって画像サイズが大きくなると通常の MSA では計算量が二次関数的に増加してしまうが Swin Transformer では Window Attention を用いることで計算量は線形に減少させることができる．

さらに Swin Transformer では振り分けられた Window を Window の半分だけシフトさせることでパッチごとに割り当てられる Window が変化し，パッチ間の相互作用を考慮した学習をすることが可能になる．

これにより Swin Transformer は計算量を削減しながら画像の特徴量を抽出することができるため，通常の Transformer よりも良いと言える．

3.3 劣化画像の生成フロー

劣化画像の超解像を深層学習で行う上で学習画像の多様な劣化表現はモデルの性能を左右する．よって本研究では劣化の種類としてぼかし，ノイズ，JPEG 圧縮，リサイズを組み合わせて劣化画像を生成する．劣化画像の生成フローを図 3 に示す．

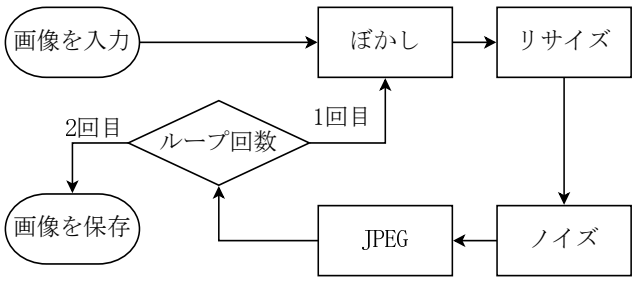


図 3 劣化画像生成フロー

3.4 ぼかし付加手順

ぼかしでは一般的にガウスぼかしが使われるが単一のぼかし手法のみではネットワークの汎化性能を上げることができない．よって本研究ではぼかしカーネルとして対称ガウスカーネル，非対称ガウスカーネル，Sinc カーネルを使用する．ぼかし画像の生成フローを図 4 に示す．

それぞれの特徴として，一般的なぼかしである対称ガウスカーネルはカーネルサイズを大きくしていくと画像全体が同じようにぼやけていく．一方，非対称ガウスカーネルではカーネルサイズを大きくしていくとカメラの手振れのような斜め方向のぼかしが生成できる．そして，Sinc カーネルではカットオフ周波数を大きくしていくと画像の輪郭部分にオーバーシュートのようなぼかしが生成できる．

ぼかしは Sinc とガウスで分岐した後，ガウスの場合は対称・非対称で一様乱数を用いた確率で分岐する．その後，畳み込みによってカーネルを画像に適用する．

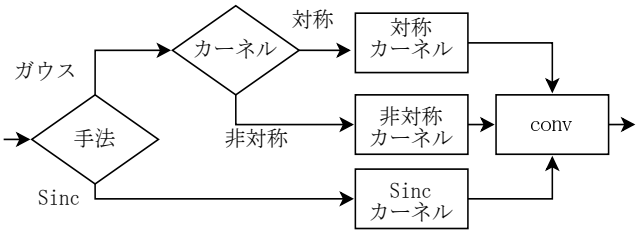


図 4 ぼかし画像生成フロー

3.5 リサイズ手順

リサイズ劣化では画像の拡大・縮小による劣化を表現するために Bicubic 補間，Bilinear 補間，Area 補間をランダムで用いて画像の拡大・縮小を行う．さらに画像のサイズ倍率をランダムで設定し倍率による劣化の度合いを変えていく．

しかし，画像のサイズをランダムに設定すると最終出力の画像が不揃いになってしまうため 2 回目の JPEG 圧縮時に目的の画像サイズにリサイズされる．

3.6 ノイズ付加手順

ノイズではガウス分布とポアソン分布を用いてノイズを生成することでよりノイズの劣化を複雑なものにする．フローの流れはまず確率でガウスノイズかポアソンノイズに分岐する．その後，それぞれ確率でカラーノイズかグレーノイズに分岐し，ノイズ画像が生成される．

ガウスノイズとポアソンノイズの違いとして，ガウスノイズは画像の画素値に関係なく確率的にノイズが付加されていくのに対し，ポアソンノイズは画像の画素値に連動してノイズの付加が変化していく．そのためポアソンノイズでは画素値が大きくなるとノイズも強くなり，画素値が小さくなるとノイズも弱くなる．

3.7 JPEG 圧縮手順

JPEG 圧縮ではより良い劣化表現を獲得するために 2 回目の JPEG 圧縮処理を変更する．2 回目の JPEG 劣化画像の生成フローを図 5 に示す．

この処理では画像がオーバーシュートの劣化後に JPEG 圧縮が行われた場合と JPEG 圧縮が行われた後にオーバーシュート劣化が発生した場合を想定している．また，sinc によるオーバーシュート劣化は確率によってスキップされる．

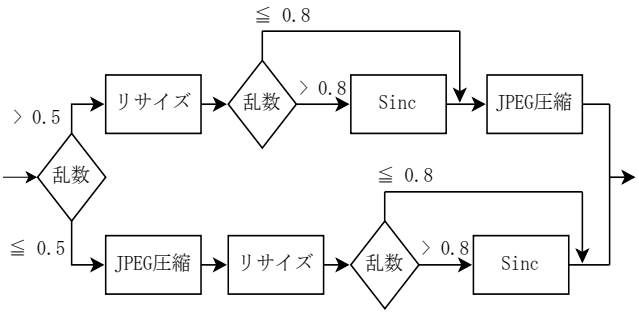


図 5 JPEG 圧縮画像生成フロー

4　実験環境

4.1 では構築環境，4.2 では使用するデータセット，4.3 では劣化画像の生成詳細，4.4 では超解像モデルの損失関数，4.5 では超解像モデルの学習設定詳細について説明する．

4.1　構築環境

本研究では Jupyter Notebook で, Python を用いてプログラムを実行する．また，ネットワークの構築，学習には Tensorflow と pytorch を使用する．

各フレームワークの特徴を表 3 に示す．Tensorflow を使用する理由は学習曲線の表示ができる TensorBoard が利用できるため，学習の監視が容易になりハイパーパラメータの調整に役立つ為である．そして Pytorch を使用する理由は深層学習の研究では Pytorch が使用されていることが多く，Pytoch を使用することで他手法のモデルの構築を用意にすることができる為である．

表 3　使用するフレームワーク

フレームワーク	特徴
Tensorflow	高，低レベルの API が使用可能 多くのプラットフォームで利用可能
Pytorch	低レベルの API が使用可能 デバックが容易

4.2　使用するデータセット

次に画像データセットの特徴を表 4 に示す．これら 4 つのデータセットを使う理由として，DIV2K と Flickr2K は様々な街の画像や日常のシーンを，OST は山や草，空などのアウトドアシーンを含んでおり，manga109 では漫画のイラストが含まれている為，これらのデータセットを使うことで様々な画像の特徴を学習することができるからである．ダウンロードした画像は 256×256 ピクセルに切り抜いて使用する．

表 4　使用するデータセットの特徴

データセット	画像の枚数
DIV2K	800 枚の画像
Flickr2K	2650 枚の画像
OST	アウトドアシーンを集めた 10342 枚の画像
manga109	109 枚の漫画の表紙画像

4.3　劣化画像生成

次にデータセットから切り抜いた画像を劣化させる．劣化を行う際のハイパーパラメータを表 5 に示す．

ぼかしの標準偏差は 1 回目と 2 回目の劣化で値の範囲を変える．ここで Cutoff はカーネルサイズが 13 以下の場合は $[\frac{\pi}{3}, \pi]$，13 以上の場合は $[\frac{\pi}{5}, \pi]$ となる．次にノイズではガウスノイズの標準偏差とポアソンノイズのスケールをぼかし同様に値の範囲を変える．図 6 の生成された劣化画像では劣化の内容が画像によって変化していることがわかる．

表 5　劣化生成パラメータ

劣化内容	項目	1 回目	2 回目
ぼかし	カーネル	[7,21] の奇数	
	回転角	[-π, π]	
	標準偏差	[0.2, 3]	[0.2,1.5]
	Cutoff	$[\frac{\pi}{3}, \pi]$	$[\frac{\pi}{5}, \pi]$
ノイズ	標準偏差	[1, 30]	[1, 25]
	スケール	[0.05, 3]	[0.05, 2.5]
	グレーノイズ	40%	
JPEG	品質係数	[30, 95]	

原画像　　劣化画像 1　　劣化画像 2

図 6　多様な劣化の例

4.4　損失関数

超解像モデルの学習で使用する損失関数について説明する．損失関数は L1 損失，知覚損失を使用する．

$\mathrm{L1_{LOSS}}$ を式 (1) に示す．ここで $\hat{x}_i$ は i 番目の正解画像，x_i は i 番目の SR 画像を表す．

$$\mathrm{L1_{LOSS}}(\hat{x}, x) = \frac{1}{n}\sum_{i=1}^{n}|\hat{x}_i - x_i| \tag{1}$$

知覚損失を式 (2) に示す．これらの特徴量は，ImageNet で事前学習された VGG19 ネットワークを使用して抽出される．VGG19 の 3 番目，8 番目，および 15 番目の層から特徴量を抽出し，知覚損失を計算する．

$$\mathrm{Perceptual_{LOSS}}(\hat{x}, x) = \mathrm{MSE}(\mathrm{VGG_{3,8,15}}(\hat{x}), \mathrm{VGG_{3,8,15}}(x))/3 \tag{2}$$

全体の損失関数を式 (3) に示す．ここで α，β，は各損失関数の係数である．

$$\mathrm{SR_{LOSS}}(\hat{x}, x) = \alpha\mathrm{L1_{LOSS}}(\hat{x}, x) + \beta\mathrm{Perceptual_{LOSS}}(\hat{x}, x) \tag{3}$$

先行研究では損失関数に L1 損失のみを使用して学習を行っていたが本研究では知覚損失を用いて学習を行う．その理由として知覚損失を利用することで出力画像がより人間の感覚に近い画像にできるためである．また知覚損失に VGG19 を使う理由として先行研究 [5] では VGG モデルが知覚損失関数として高い性能を発揮しながらも使

用するメモリ容量が同程度の性能を発揮する知覚モデルよりも低いため VGG モデルを選択することを推奨しているためである．

4.5　学習設定詳細

ネットワークのハイパーパラメータについて STB の数を 4，Window サイズを 8，パッチサイズを 4，埋め込み次元を 180，MLP の層の数を 2，CNN のフィルタサイズを 3，損失関数の (α,β) をそれぞれ (1，0.005) とした．

学習時の設定については最適化手法を Adam，学習率を 0.0002，バッチサイズを 16，エポック数を 40 とした．また，GPU は RTX4090 を使用した．

5　実験結果

5.1 では DIV2K 検証用データセットでの性能比較，5.2 では Set14 データセットの超解像結果，5.3 ではブラインド超解像ついて説明する．

5.1　DIV2K 検証用データセットでの性能比較

それぞれのモデルの PSNR と SSIM の平均値を表 6 に示す．表より，本研究の提案手法は先行研究の SwinIR と比較して PSNR では約 8.5% ，SSIM では約 5.6% 向上している．

表 6　DIV2K での超解像性能

モデル	PSNR(dB)	SSIM
SwinIR	28.75	0.8501
ours	31.43	0.9010

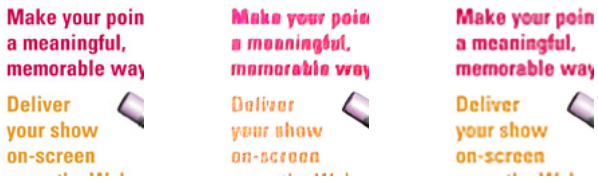
5.2　Set14 データセットの超解像結果

それぞれのモデルの PSNR と SSIM の平均値を表 7 に示す．表より，本研究の提案手法は SwinIR と比較して PSNR では約 6.7% ，SSIM では約 5.8% 向上している．

表 7　Set14 での超解像性能

モデル	PSNR(dB)	SSIM
SwinIR	26.96	0.8043
ours	28.92	0.8541

また，出力された画像を見ると，SwinIR では画像内の記されている文字を読むことが難しいほど崩れている．一方，提案手法では文字を読むことが可能な程度まで復元されている．



HR　SwinIR　ours

図 7　Set14 の超解像結果

5.3　ブラインド超解像結果

正解画像が無い超解像データセットを用いて比較を行う．SwinIR ではレンガ造りの壁が復元できておらず，隣のレンガ同士がくっついたような表現になってしまっている．一方，提案手法ではレンガ造りの壁を表現しながらも LR に比べてノイズも無く，精細な画像になっている．

LR　SwinIR　ours

図 8　ブラインド超解像結果

6　結び

本研究では SwinIR をベースに劣化を組み合わせた画像を学習することで現実世界の劣化画像に対応した超解像モデルを作成した．その結果，ベースモデルよりも DIV2K では PSNR は約 8.5%，SSIM は約 5.6% 向上し，Set14 においても PSNR は約 6.7%，SSIM は約 5.8%向上した．

今後の課題は超解像画像全体がぼやけてしまっており，シャープな輪郭の表現ができていないのでシャープマスクを学習画像に適応させるなどをすることでこの問題を解消していきたい．

参考文献

[1] Fortune Business Insights,“衛星画像市場の成長，規模，シェア，および地域予測，2023〜2030 年,” https://www.fortunebusinessinsights.com/satellite-imaging-market-103372, 参照 Jun. 2023.

[2] Xintao Wang et al.,“Real-ESRGAN: Training Real-World Blind Super-Resolution With Pure Synthetic Data,” IEEE/CVF International Conference on Computer Vision Workshops, pp.1905-1914, Oct. 2021.

[3] Jinsu Yoo et al.,“Enriched CNN-Transformer Feature Aggregation Networks for Super-Resolution,” IEEE/CVF Winter Conference on Applications of Computer Vision, pp.4956-4965, Waikoloa, Hawaii, Jan. 2023.

[4] Jingyun Liang et al.,“SwinIR: Image Restoration Using Swin Transformer,” IEEE/CVF International Conference on Computer Vision Workshops, pp.1833-1844, Oct. 2021.

[5] Gustav Grund Pihlgren et al.,“A Systematic Performance Analysis of Deep Perceptual Loss Networks: Breaking Transfer Learning Conventions,”Computer Vision and Pattern Recognition, arXiv:2302.04032, Oct. 2023.