要旨

AI による画像生成技術が進化しており，人間が見て従来の画像と区別がつかない画像を生成できるようになっている．その結果，画像の信頼性や創作性の検証が難しくなり，著作権問題やフェイク画像などの AI 生成画像を巡るトラブルが発生している．そのため，AI 生成画像を高精度で検出し，真偽の判定を可能にする技術の開発が重要な課題となっている．本研究では，深層学習を用いることなく，AI 生成画像と人が描いたイラスト画像を区別する手法を開発し，その精度と汎用性を向上させることを目的とする．この技術により，創作物の透明性の向上に貢献することを目指す.

本論文では“Stable Diffusion”で生成した画像に関して判別を行う．この生成方法は潜在拡散モデルを用いており，ノイズデータから任意の画像が生成される．ノイズデータを元に画像が生成されるため，隣接画素の整合性が十分に保証されないと考えられる．そのような性質を用いて判別を行う方法として，画像の隣接画素の差分を求めるハールウェーブレット変換を用いた.

ハールウェーブレット変換は，画像を 2×2 ブロックに分割し，横差分・縦差分・斜め差分を求める計算手法である．この計算では，2×2 ブロックに分割する際の開始位置をずらすことで 4 通りの計算が生じる．この 4 通りそれぞれにハールウェーブレット変換を行い，横差分・縦差分・斜め差分を計算する．この結果として得られる計 12 通りの画像を元に判別を行う．判別の方法として，まずハールウェーブレット変換後の各画像の全画素値の出現回数を集計する．その後，横差分，縦差分，斜め差分ごとにまとめた 4 枚の画像について，2 枚ずつのペアの組み合わせ($_4C_2$=6 通り)を作成する．作成したペアごとに同じ画素値の出現回数の差の絶対値を全画素値に対して計算し，その合計値を画像サイズ(縦×横)で割り，100 を掛ける．さらに，この結果から得られた値を用いてペアを作り($_6C_2$=15 通り)，各ペア間の差の絶対値を計算する．この 15 個の値の中の最大値を，計算に用いた画像の差分(横差分か縦差分か斜め差分)の評価値とした.

この評価値を用いて AI 生成画像と人が描いた画像を比較した．結果として，AI 生成画像と人が描いたイラスト画像を判別する際に，横差分と斜め差分の精度はかなり高くなるが，縦差分の精度は落ちるという結果を得た．しかし，どの差分の場合でも統計的に有意な差は出ている．一方で，StableDiffusion 以外の画像生成 AI を用いた場合は，StableDiffusion よりも精度が落ちるため，判別が難しいという結果を得た.

目次

第1章　　序論

1.1 背景

従来の画像は，現実世界の人や動物，風景を写真撮影した画像(本論文ではリアル画像と記述する)や，パソコンやペンタブレット等の描画ソフトを使って描いたアニメや漫画・ゲーム風の画像(本論文ではイラスト画像と記述する)などである．それに対して，近年ではこれらと区別がつかない画像を，AI で生成することが可能となっている．AI を用いて生成した画像(本論文では AI 生成画像と記述する)に関する技術は日々進化しており，人間が見分けることが困難になっている．その結果，画像の信頼性や創作性の検証が難しい状況が発生している．そのため，AI 生成画像を高精度で検出し，真偽の判定を可能にする技術の開発が重要な課題となっている．

AI 生成画像は膨大な画像データを元にして生成されているが，画像データのなかには著作権者に無断で使用されている画像が多数存在しており，AI を学習させる段階での著作権問題が発生している．また，特定の作者の絵柄に寄せた画像を出力することも可能となっており，そのような生成画像は著作権侵害にあたる可能性がある[1]．また自身が描いたイラスト画像を AI 生成画像と決め付けられ，誹謗中傷などにより気分を害する問題も多発している[2]．また，AI 生成画像を自分が描いたイラスト画像と偽る行為やイラスト画像を描く仕事をしている人の仕事を奪うといった問題，イラスト画像のコンテストにおいて AI 生成画像を提出し受賞したり，審査を突破したりするといった問題も実際に起きている[2]．

現在，リアル画像・イラスト画像問わず，AI 生成画像かどうかを判別する方法として，深層学習を用いる場合が多い．しかし，深層学習を用いるためには既存画像を学習させる必要があり，AI 生成画像の問題点として挙げた著作権の問題点と同様の問題が発生する可能性がある．

このような点から本研究では，イラスト画像とそれを模した AI 生成画像に関して，深層学習を使用しないで判別するための研究を行なった．

1.2 目的

本研究の目的は，AI 生成画像と人間が作成したイラスト画像を区別する手法を開発し，その精度と汎用性を向上させることである．AI 生成画像を適切に判別する技術を開発することにより，デジタルコンテンツの創作性，透明性の向上に寄与することができる．これにより，著作権侵害の抑制に資する技術の発展につながる．

なお本研究はイラスト画像と AI 生成画像の判別に関する研究をしているが，AI 生成画像を悪者とする意図は一切ない．

1.3 論文構成

本論文は，第 2 章で，AI 生成画像の生成方法について説明する．第 3 章では，本研究で AI 生成画像の検出を行うための方法の説明や作成したプログラムの動作について説明する．第 4 章では作成したプログラムを実行した結果を提示し，判別機能を評価する．第 5 章で，本研究に関する結論を記述する．

第2章　　AI 生成画像の基本技術

2 章では，AI 生成画像の生成原理について説明する．画像生成 AI のツールとして著名なものに，“Stable Diffusion”，“NovelAI”，“Midjourney”，“Dall-E3”がある．ここでは，Stable Diffusion について説明する．

2.1 Stable Diffusion における画像生成の原理

Stable Diffusion はプロンプトとして文字情報を入力し，その文字情報を表現するような画像を生成する．画像を生成する部分の深層ニューラルネットワークは，潜在拡散モデル[3]を用いて学習させている．図 1 に Stable Diffusion の画像生成時の挙動，図 2 に Stable Diffusion のトレーニング時の挙動を示す．以下では，これらの図を用いて，Stable Diffusion の動作を説明する[4]．

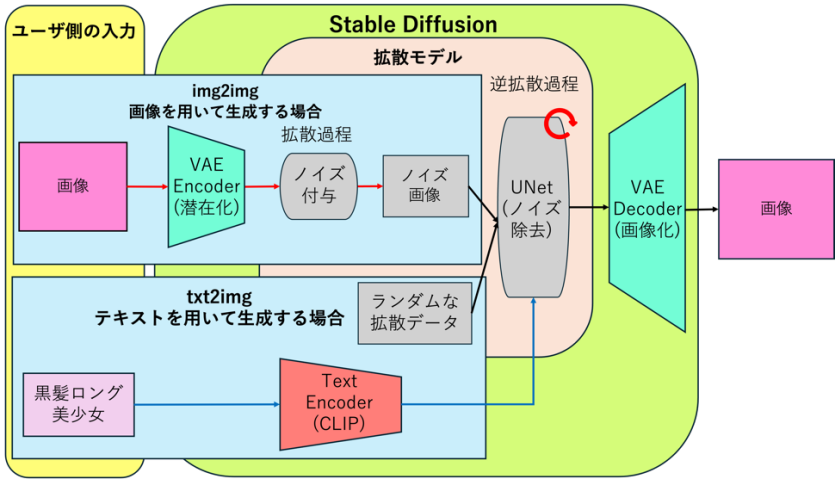


図 1 Stable Diffusion の画像生成時の挙動

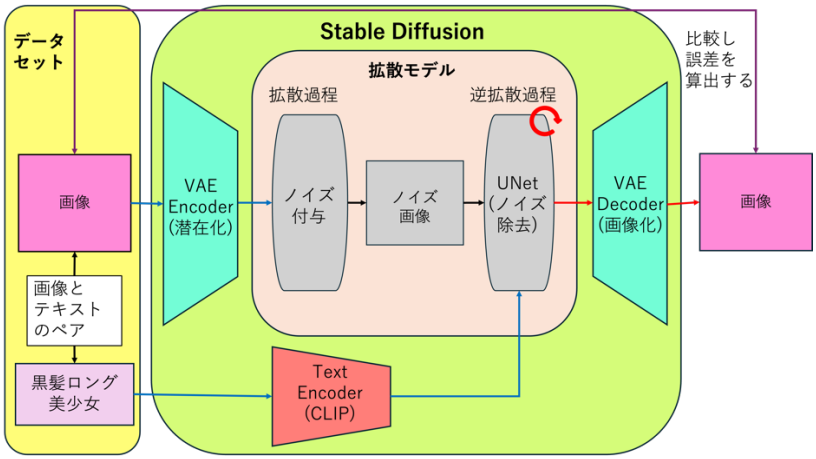


図 2 Stable Diffusion のトレーニング時の挙動

2.1.1 Stable Diffusion を構成する各要素の機能

テキストエンコーダ（CLIP）は入力された文章を埋め込みベクトルと呼ばれる数値表現

に変換する役割を持っている．この埋め込みベクトルはテキストの意味や構造を表現したものであり，これによりユーザが入力したテキストに沿った意味を指定することができる．Stable Diffusion は，CLIP のテキストエンコーダ機能を活用することで，ユーザが入力したテキストから，高い精度で多様な内容の画像を生成できるようになっている[5]．CLIP は画像と言語の関連性を学習するためのモデルである．このモデルではテキストと画像をペアとして扱い，それぞれをベクトル表現に変換し，特徴空間へマッピングする．この過程を通じて，テキストと画像の意味的な関係を抽出し，関連性が高いペアほど高いスコアを付与するようにトレーニングが行われる．トレーニングにより得られたテキストのベクトル表現はテキストエンコーダに，画像のベクトル表現はイメージエンコーダに保存される．これを活用した LAION データセットは，web 上から自動収集された 55 億組以上のカラー画像とテキストのペアのデータセットであり，CLIP モデルによる処理が施されたデータが含まれている．このデータセットは，非常に広範囲かつ多様なデータセットとなっている[5]．

VAE エンコーダーは，データセットを使った学習や画像を用いて画像を生成する際に重要な役割を果たしている．VAE エンコーダーは入力された画像を潜在空間上の表現(潜在ベクトル)に変換する．この潜在ベクトルは，拡散モデルが動作するための基盤となる情報を提供する．

拡散モデルは大きく拡散過程と逆拡散過程の 2 つに分かれている．拡散過程では，画像データに対してランダムなガウスノイズを少しずつ付与していき，最終的に完全なノイズ画像にする．このプロセスでは，画像の情報を分散させることで，学習におけるノイズ除去能力を強化する役割を果たす．一方，逆拡散過程では，拡散過程で付加されたノイズを少しずつ数百~数千回のステップを経て除去しながら，元の画像に近づけていく．この過程には，U-Net と呼ばれる畳み込みニューラルネットワークが用いられる．U-Net はトレーニング時にはノイズ画像を入力として受け取り，入力画像から取り除くべきノイズを出力するように学習される．画像生成時には，ガウスノイズを初期状態として，U-Net が出力する除去対象のノイズを入力サイズから引くことで，ノイズを徐々に低減させる．このプロセスを複数回繰り返すことで最終的な画像を生成する．さらに，CLIP によって作成された特徴ベクトルを逆拡散過程の各ステップでガイドとして機能させることで，テキスト入力の意味内容に合致した画像生成が可能となる．これにより，ユーザが指定したテキスト情報を忠実に反映した画像を生成する能力が向上する．拡散モデルは，入力された画像のデータ空間で計算を行うが，潜在拡散モデルはデータを圧縮して低次元に変換した空間である潜在空間で計算を行う．これにより計算コストが減り，画像生成の効率が向上する．

VAE デコーダーは，逆拡散過程で収束させた最終的な潜在空間上のデータを，人間が識別可能な画像データに再構成する役割を担っている．これにより画像データをピクセルデータとして出力することができる．

2.1.2 Stable Diffusion の画像生成時の挙動

図 1 は Stable Diffusion の画像生成時の挙動の模式図である．Stable Diffusion には txt2img と img2img の二つの生成方法がある．

img2img は，ユーザが入力した画像を基に新たな画像を生成する方法である．このプロセスでは，まず入力された画像が VAE エンコーダによって潜在空間上のデータに変換され，そのデータが拡散モデルに送られる．拡散モデル内部では，拡散過程と逆拡散過程を経てデータが生成される．最終的に，得られた潜在データは VAE デコーダによって再構成され，最終的な画像が生成される．

一方 txt2img は，ユーザが入力したテキストを元に画像を生成する方法である．このプロセスでは，ランダムに設計されたノイズを元に画像が生成される．入力されたテキストはテキストエンコーダによって処理され，得られた特徴表現とランダムノイズが事前学習済みの逆拡散過程に入力される．その後，逆拡散過程で生成された潜在データは VAE デコーダーにより再構成され，最終的な画像が生成される．

2.1.3 Stable Diffusion のトレーニング時の挙動

図 2 は Stable Diffusion のトレーニング時の挙動の模式図である．Stable Diffusion のトレーニング時には U-Net(逆拡散過程)のトレーニングを行なっており，VAE や CLIP はあらかじめ学習済みのものを用いる．まずデータセットにある画像とテキストのペアを入力で受け取る．画像データは VAE エンコーダによって潜在空間上の表現に変換された後に拡散モデルに入力され，その変換後の値に対して拡散過程と逆拡散過程が適用される．一方，テキストデータはテキストエンコーダを用いて特徴量に変換され，生成過程で逆拡散過程に統合される．ここでの逆拡散過程におけるノイズの除去を，U-Net にトレーニングさせる．このプロセスで得られた潜在データは，最終的に VAE デコーダによって再構成され，画像として出力される．その画像を元の画像と比較して再構成損失を計算する．この損失をもとにモデルを調整し，元画像にできるだけ近い再構成が可能となるようにトレーニングしていき，精度を向上させる．

2.2 AI が生成したイラスト画像の性質

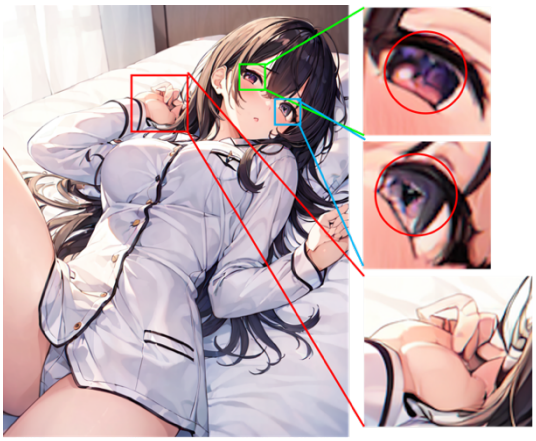
AI が生成したイラスト画像の性質として，生成された画像の細部が不自然になる場合がある．これらの中で，判別に利用できそうな特徴として以下のようなものがある[6].

① 左右の瞳の形状やハイライトが一致しない.

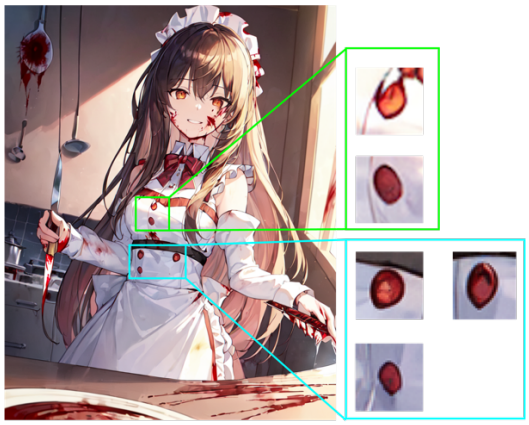
② 一列に並ぶボタン列のような規則的な反復形状において一貫性が不十分である.

③ 指や髪などの末端の描画が他の物と融合してしまう.

以下の図 3(a)~(c)はその例である.



(a) AI 生成画像(付録 1)



(b) AI 生成画像(付録 2)

(c) AI 生成画像(付録 3)

図 3 AI 生成画像とその画像での違和感の例

図 3(a)の緑枠で囲った右目と青枠で囲った左目を比べると瞳の形やハイライト，目の色などが左右で異なっており，瞳は歪な形になっている．特に赤丸で囲んだ部分を見比べるとかなり形状が異なっている．また赤枠で囲った右手の形が変な方向に曲がっていたり，指が繋がっていたりする．他にも指の本数が 5 本以外で出力されたり，指の形がおかしく出力されたりすることもある．図 3(b)の緑や青枠で囲った部分のボタンの形が綺麗な円で出力されておらず，それぞれの形も一致していない．また青で囲んだボタンは右が 1 個なのに対し左ボタンが 2 個となっている．これは意図したデザインの可能性もあるが違和感がある．図 3(c)の赤枠で囲った部分は，髪の毛と右手が融合している．また緑丸で囲った部分は，指の輪郭の線が髪の毛の輪郭の線になっている．

拡散モデルを用いて出力した AI 生成画像に歪みが生じる要因として，2．1 で説明したようにノイズから画像を生成する仕組みに起因していると考えられる．拡散モデルでは，初期のランダムノイズから徐々に意味のある画像を復元するため，モデルは画像の全体的な構造や大まかな特徴をトレーニングすることに焦点を合わせている(手や目など特定の対象について焦点を合わせているものも存在する)．この過程で，モデルは画像を統計的なデータとして学習しているため，その物体が何なのかといった個々の要素が持つ意味を理解している

わけではない[7]．これにより，画像の大まかな特徴(人体の構造など)は比較的うまく再現される一方で，細部の表現（例えば目の対称性や形が一貫する装飾など)においては，生成過程で加えられるノイズの影響が働き，出力画像において整合性を保つことが困難となる．この問題は，拡散モデルが画像の細部の描写や理論的背景，機能的役割を十分に理解することが難しいことに起因すると考えられる．特に複雑な構造や多様な要素が絡み合う場合，生成される画像には歪みや不自然な表現が現れやすい．その理由は，拡散モデルが訓練データ基づき，画像の統計的特徴を再現することに重きを置いているため，理論的背景や機能的役割に基づく直観的な判断が欠けているためである．

図 3(a)では「目の左右対称性（塗りや形状がほぼ同じであり，瞳が正円に近い形状を持つ)」といったイラスト描画における理論をモデルが十分に捉えられず，左右の目の形状や塗りが一致せずに出力されていることに現れている．ただし，生成モデルは大量のデータを元に学習を行っているため，出力は概ね規則に沿った目を生成する傾向があるが，細部において歪が生じる．同様に，手の形状についても，「指は基本的には 5 本あり，それぞれが一定の形状を持つ」という基本的な原則を完全には捉えられておらず，不自然な形状の手が出力されている．しかし，近年の生成モデルは性能が向上しており，特に適切なプロンプトを記述することで，初期の AI 生成画像において頻発していた手の形状の歪みを大幅に軽減し，より正確に描写できるようになっている．

図 3(b)では，「同じパーツを一貫した図形として出力することや，綺麗な幾何学的形状(正円や正方形など)を描写することが苦手」といったことが，5 つあるボタンの形が微妙に違うことや正円とは違い若干歪んでいることに現れている．これは潜在空間の生成プロセスが全体的な調和を重視するため，局所的なピクセル精度（正円や正方形）を保証するメカニズムが弱いためである．またモデルは，訓練時に学習したデータから幾何学的形状や構造の類似性を再現しようとしており，厳密な形状や類似した形状の再現を行っているわけではないそのため，異なる部分の形状や構造を動的に生成する過程で，各要素の相対的な位置関係や形状がわずかに変化することが避けられないと考えられる．

図 3(c)では，髪の毛と指の融合や髪の毛の輪郭線と指の輪郭線の融合が生じており，「一つの線が担う役割は基本的に一つである（つまり,物体 A の境界を描いていた線が,急に物体 B の境界を描いた線に変わることがない.)」といった基本的な性質が守られていない．これは，潜在拡散モデルは複数の要素を込み合わせて画像を生成するため，異なる物質や形状を融合させた中間的な結果を生成するためである．また，潜在拡散モデルは局所的な情報(ピクセル周辺のパターン)を元に画像を生成するため，全体の構造や意味を完全に把握できていない．そのことから，複雑な構造や細かいディテールを要求される場合に，モデルが適切な境界や構造を再現する能力がうまく発揮できなくなり，不自然な形状や融合が発生することがある．

第3章　　本研究における AI 生成画像の判別方法

2．2 節の内容のうち，図 3(a)のケースのように瞳の形状や色を判断材料として用いる場合，描写する目の範囲や角度の違い，髪や睫毛による一部の隠蔽，さらにはオッドアイ（左右の眼で虹彩の色が異なること）などの特殊な要素を考慮する必要がある．しかし，これらの要素を精度良く判別することは困難である．図 3(b)に示すケースでは，複数の同一形状が自然に並んだ部分のみ(例えばボタンや模様)を正確に抽出して判別することや，逆に，意図的に形状が変えられたものを区別する必要がある．しかし，これらを正確に解析することは難しい．図 3(c)のケースでは，何が正常で何が異常かの判断が難しく，その部分(図 3(c)の場合では，髪と肌が融合した部分)をプログラムで抽出すること自体が困難である．これらの AI 生成画像における不自然さは，視覚的に判別可能であっても，手書きによる修正や AI を用いた加工によって容易に改変することが可能である．以上のことから，本研究では，視覚的特徴に基づく判別ではなく，目では識別できないような差異を画素値の統計的な解析によって抽出し，AI 生成画像かイラスト画像かを判別したいと考えた．

AI 生成画像に感じる違和感の一つに色の塗りがある．これは私の主観であり，客観的に説明することが難しいが，AI 生成画像は独特の塗りをしていることが多いと感じた．また，Stable Diffusion の生成画像はノイズから生成しているため，一定の塗りが続く場所やグラデーションになっている部分において，微妙な画素値の揺れが生じると考えた．

(a) AI 生成画像(付録 4)

(b) 四角で囲んだ部分の拡大

(c)図 4(a)で最も多い画素値の表示

図 4　AI 生成画像と画素値の最頻値

図 4(a)は AI 生成画像であり，その画像で最も多い画素値は(R，G，B)=（253，242，231）で 1127 個となっている．その画素値が多い部分を赤四角で囲み，図 4(b)に切り取る．また図 4(b)の最も多い画素値をそのまま表示し，それ以外の画素値を黒で表示したものを図 4(c)に示す．図 4(c)を見ると，　最も多い画素値の内部に黒色部分が不規則に現れている．それは塊状になっていることもあり，ポツポツと現れることもあり，一定の塗りが続いてい

ないことがわかる．

このような特徴を画素値の解析を用いて判別する際に，私はハールウェーブレット変換を用いるとAI生成画像とイラスト画像に差異が生まれると考えた．ハールウェーブレット変換を採用した理由は，この手法が隣接画素間の差分を算出する点である．これにより，一定の塗りが続く領域やグラデーションになっている領域において，画素値のわずかな違和感を数値化できると考えた．

3.1 ハールウェーブレット変換

3．1．1 ハールウェーブレット変換の計算

ハールウェーブレット変換は，ウェーブレット変換のなかで最もシンプルな手法である．主に信号や画像の特徴を階層的に分解し，効率よく処理や分析を行うために使われる．ハールウェーブレット変換を用いると，データを局所的なスケール(詳細)と全体的なスケール(大まかな傾向)に分解することができる．ハールウェーブレット変換は，以下のステップで処理する．式(1)~式(3)は[8]を引用して記述している．

一次元のデータ$(x_1,\ x_2)$に対して$\left(\frac{x_1+x_2}{2},\ \frac{x_1-x_2}{2}\right)$を対応させる写像をハール変換と呼ぶ．

この計算を行列で表すと

$$(x_1,\ x_2)\ \mapsto (x_1,\ x_2)\begin{pmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & -\frac{1}{2}\end{pmatrix}\ \cdots(1)$$

と表せる．式(1)を計算すると

$$(x_1,\ x_2)\ \mapsto\ \left(\frac{x_1+x_2}{2},\ \ \frac{x_1-x_2}{2}\right)\ \cdots(2)$$

となる．式(2)において$\frac{x_1+x_2}{2}$はx_1とx_2の平均であり，$\frac{x_1-x_2}{2}$はx_1とx_2の差分である．二次元のデータに対して計算を行う場合は，式(1)の計算を横方向に適用した後，縦方向に適用し

$$\begin{pmatrix}x_{1,\ 1} & x_{1,\ 2}\\ x_{2,\ 1} & x_{2,\ 2}\end{pmatrix}\mapsto\begin{pmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & -\frac{1}{2}\end{pmatrix}\begin{pmatrix}x_{1,\ 1} & x_{1,\ 2}\\ x_{2,\ 1} & x_{2,\ 2}\end{pmatrix}\begin{pmatrix}\frac{1}{2} & \frac{1}{2}\\ \frac{1}{2} & -\frac{1}{2}\end{pmatrix}$$

$$=\begin{pmatrix}\frac{(x_{1,\ 1}+x_{1,\ 2})+(x_{2,\ 1}+x_{2,\ 2})}{4} & \frac{(x_{1,\ 1}-x_{1,\ 2})+(x_{2,\ 1}-x_{2,\ 2})}{4}\\ \frac{(x_{1,\ 1}+x_{1,\ 2})-(x_{2,\ 1}+x_{2,\ 2})}{4} & \frac{(x_{1,\ 1}-x_{1,\ 2})-(x_{2,\ 1}-x_{2,\ 2})}{4}\end{pmatrix}\rightarrow\begin{pmatrix}A & Dl\\ Dv & Dd\end{pmatrix}\ \cdots(3)$$

となる．式(3)においてAは4つの値の平均であり，Dl(difference lateral)は横方向の差分，Dv(difference vertical)は縦方向の差分，Dd(difference diagonal)は斜め方向の差分となっている．この計算を画像データに適用する．画像データは，一般的に(横，縦，RGB値)の三次元のデータとなっているが，本研究ではRGB値をそれぞれのチャネルに分けて計算を行う．このため，(横，縦)の二次元データに対してRGBの各チャネルで計算を三回実

施し，その結果を統合して最終的な画像を構成する．画像データを対象として，式(3)による計算を適用するために，画像を 2 画素×2 画素ずつに分割する．次に分割された 2 画素×2 画素の値に対して式(3)を適用し，得られた各値 A・Dl・Dv・Dd をそれぞれ集約した行列を構築する．

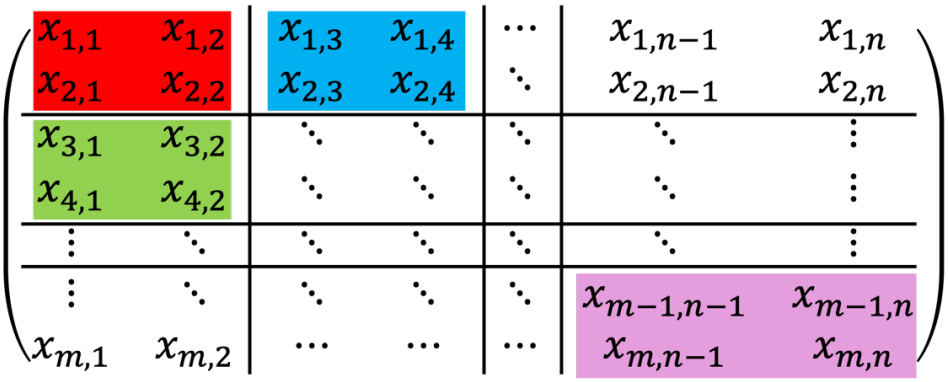


図 5 画像データの行列

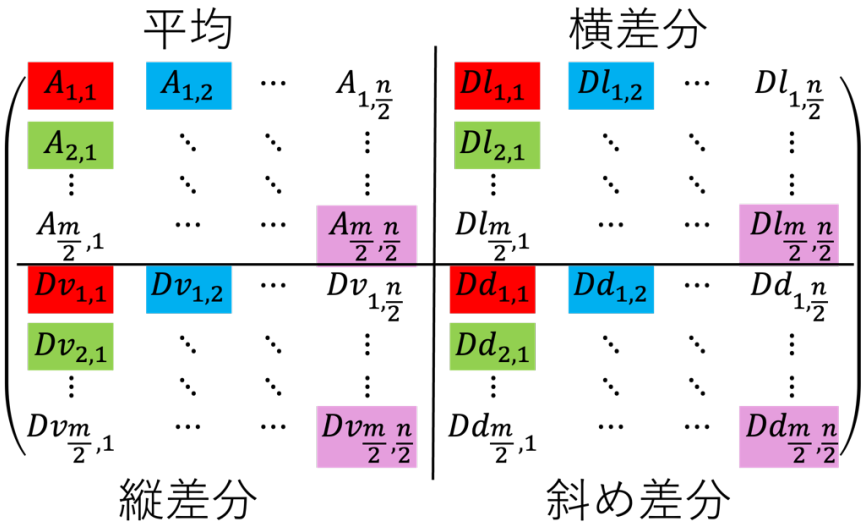
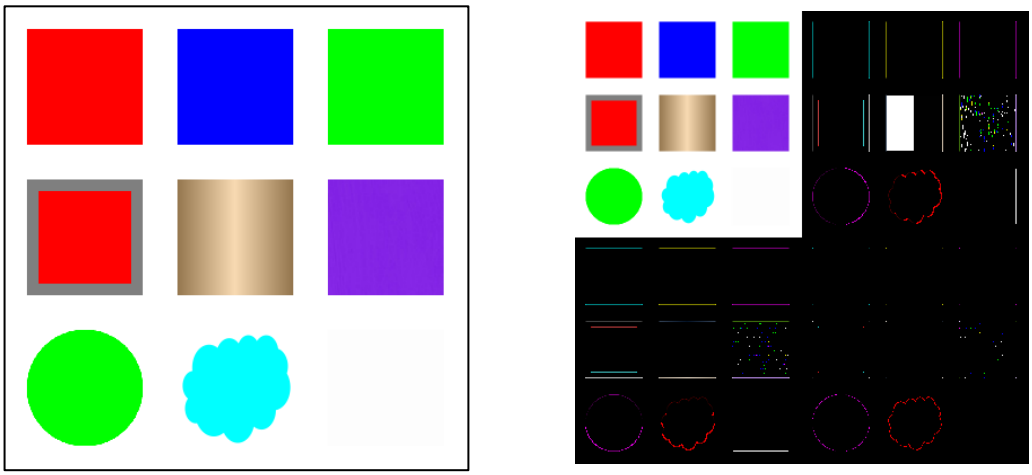


図 6 画像データに式(3)を適応した後の行列

図 5 は画像データを行列として表現したものである．ここで m と n はそれぞれ画像データの高さと幅を表している．画像データに対して，2 画素×2 画素ごとに，A・Dl・Dv・Dd を算出するため，A・Dl・Dv・Dd の個数は画像データの縦横の半分になる．行列の要素 $x_{i,j}$ は対応する画素の R，G，B 値を表している．図 6 は図 5 に式(3)を適用した結果得られた行列を表している．ここで，図 5 と図 6 の背景色が一致する部分が対応している．例えば，図 5 において，背景を赤色で示した 2×2 行列に式(3)を適用すると，図 6 において，背景を赤色で示した A・Dl・Dv・Dd の値が計算できる．

これを実画像に適用した例を図 7(a)(黒枠線は見やすいように後付けしたものであるため，計算には影響しない)，図 7(b)を用いて説明する．



(a) 実画像の例　　(b) 図 7(a)に対して式(3)を適用した結果

図 7 図 5 と図 6 を実画像で行った結果

図 7(a)は説明に用いるために用意した 1 枚の画像で，9 箇所の正方形領域を図のように塗りつぶした画像である．各領域の画素値は，赤色が(R，G，B)=(255，0，0)，緑色が(R，G，B)=(0，255，0)，青色が(R，G，B)=(0，0，255)，灰色が(R，G，B)=(128，128，128)，水色が(R，G，B)=(0，255，255)となっている．中段の左の正方形は，上段ではわかりにくかった赤→別色と別色→赤の違いを示すために用意した．真ん中のグラデーションは，中央まで RGB の値をそれぞれ 2 ずつ加算し，中央以降は 2 ずつ減算したものになっており，グラデーション時の出力を示すために用意した．紫色の部分は一見すると RGB の値が同じように見えるが，微妙に画素値がばらついており，そのばらついている際の出力を示すために用意した．下段の円と雲の形は，上中段でわかりにくかった曲線の出力を示すために用意した．右下の四角は(R，G，B)=(253，253，253)となっており，背景とのわずかな画素値の差の出力を示すために用意した．

図 7(b)は図 7(a)の画像をハールウエーブレット変換した画像である．図 7(b)は，左上以外の大部分が黒色になっている．その理由は，隣接する画素の値が同じ場合，*Dl*，*Dv*，*Dd* の値は$\frac{(i-i)+(i-i)}{4}=0$，$\frac{(i+i)-(i+i)}{4}=0$，$\frac{(i-i)-(i-i)}{4}=0$(i は画素値)となり，画素値(R, G, B)=(0, 0, 0)と，黒色になるためである．図 7(b)を見ると，*Dl* は横差分を計算したものであるため，横に隣接した画素値に変化がある場合に黒色以外の値が出力されている．*Dv* は縦差分を計算したものであるため，縦に隣接した画素値に変化がある場合に黒色以外の値が出力されている．*Dd* は斜め差分を計算したものであり，図 7(a)にある正方形ではあまり変化は見られない．しかし直線ではない円の場合は *Dl*，*Dv* より変化が見られる．

次に，実際の計算の例を詳細に示すために，図 7 (a)，(b)の一部分を拡大した下記の図の場合における計算結果を説明する．

(a)灰色→赤の境目(黒線は後付け)　　(b)赤→灰色の境目(黒線は後付け)

図 8 図 7(a)(b)の一部を拡大

図 8(a)の場合，*Dl* は(R，G，B)=($\frac{(127-255)+(127-255)}{4}$，$\frac{(127-0)+(127-0)}{4}$，$\frac{(127-0)+(127-0)}{4}$) = (−64，63．5，63．5)となる．開発したプログラムは Python の“uint8”で計算を行っており，結果がマイナスの値になった場合はその値に 256 を足した値が出力となる．そのため，

“-64+256=192”となる．また，小数点以下は切り下げとなる．そのため，出力結果は(R，G，B)=(192，63，63)となり，えんじ色に近い色が出力される．また，

Dv は(R，G，B)= ($\frac{(127+255)-(127+255)}{4}$，$\frac{(127+0)-(127+0)}{4}$，$\frac{(127+0)-(127+0)}{4}$) = (0，0，0)，

Dd は(R，G，B)= ($\frac{(127-255)-(127-255)}{4}$，$\frac{(127-0)-(127-0)}{4}$，$\frac{(127-0)-(127-0)}{4}$) = (0，0，0)

となるので黒色が出力される．一方図 8(b)の場合，

Dl は(R，G，B)=($\frac{(255-127)+(255-127)}{4}$，$\frac{(0-127)+(0-127)}{4}$，$\frac{(0-127)+(0-127)}{4}$) = (64, −63.5, −63.5)

となり，(R，G，B)=(64，193，193)の青緑色に近い色が出力される．このようにして計算が行われているため，図 7(a)中央のグラデーションの領域は，隣接画素の差分の絶対値は変わらず，左半分はプラス，右半分はマイナスになる．このため，*Dl* に対応した領域で，左半分が白色，右半分が黒色となる．また，紫の正方形の出力では，隣接する画素値に変化が生じているため，黒色以外がまばらに出ている．

Dd の出力結果は正方形に対しては四隅だけが黒色以外の出力になる．*Dd* は斜め差分成分であるため曲線が多い円や雲の形の輪郭がはっきりと出力される．

3．1．2 ハールウェーブレット変換の画像への適用

本研究で行なった計算を図 9 を用いて説明する．画像サイズは幅が 768(pixel)，高さが 960(pixel)のもので説明する．このサイズは，Stable Diffusion のサイズの初期値である 512*640 を，それぞれ 1．5 倍にしたものである．

図 9 AI 生成画像(付録 5)

本研究では，ハールウェーブレット変換の計算を行いやすいように画像を正方形にする．足りない部分を追加する場合は，(R，G，B)=(255，255，255)の色を追加する．この際，後々の計算のために画像の幅と高さの大きい方が奇数の場合は 3pixel 大きくし，偶数の場合は 2pixel 大きくする．そして，幅と高さの小さい方は大きい方と同じサイズになるように(R，G，B)=(255，255，255)の色を追加する．これを図 9 に適用すると図 10 のようになる．

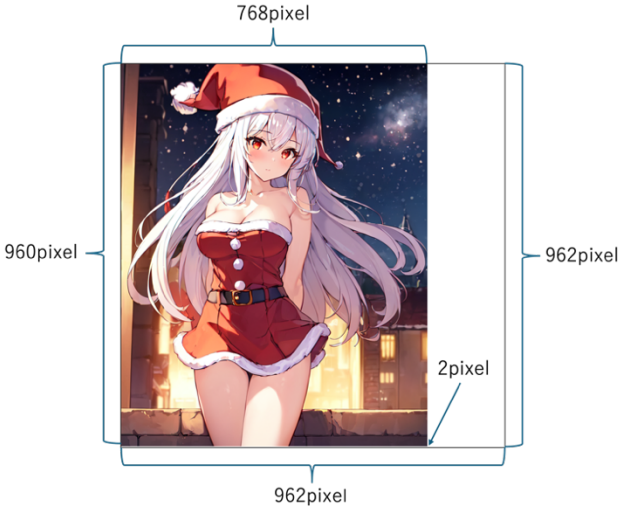


図 10 サイズを変更した画像（この画像は下辺に 2pixel 幅の白領域を追加した）

ウェーブレット変換は画像を 2×2 のブロックに分割して計算するため，分割する行列の始点(図 5 の赤色背景の部分にあたる 2×2 のブロック)の範囲を以下の図 11 のように横に一マス，縦に一マス，縦横に一マスそれぞれずらせば計算結果も変わってくる．2 マス以上ずらした場合は，4 通りのどれかと同じ結果になる．

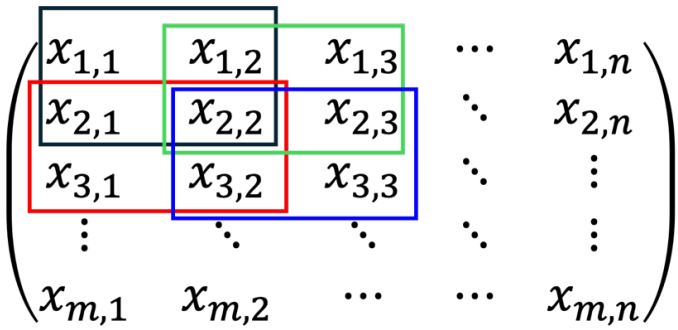
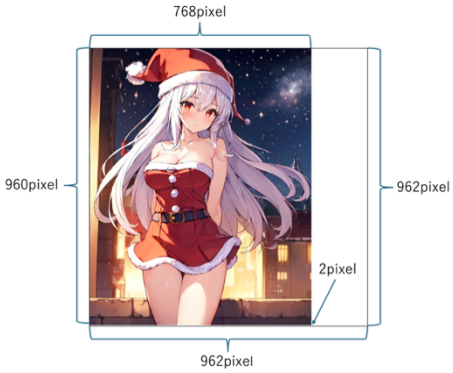
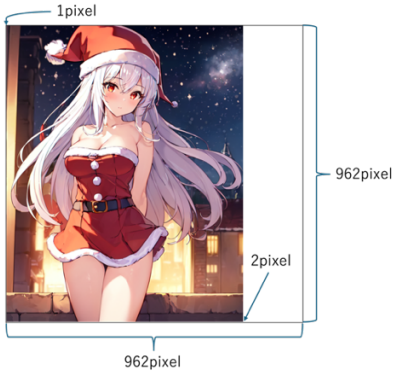


図 11 行列の範囲のずらし方

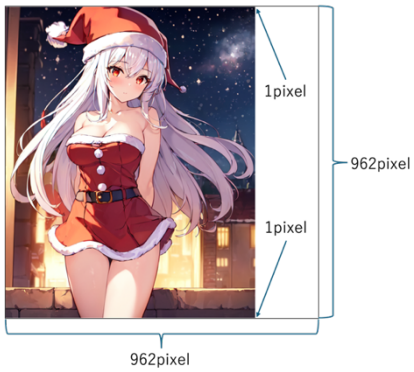
図 11 を図 10 に適用したものを図 12 にまとめる．図中で 2pixel，1pixel と示した箇所はその幅の領域を白色で置き換えている．黒線は後付けしたものである．



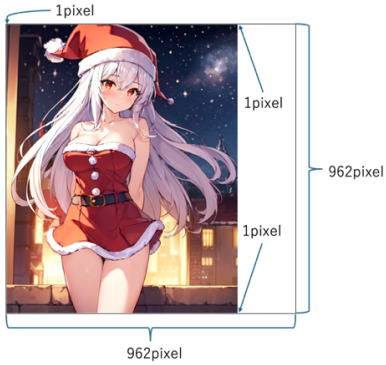
(a) 図 11 での黒枠が始点の時
（これ以降は“00”と記述する）



(b) 図 11 での緑枠が始点の時
（これ以降は“10”と記述する）



(c) 図 11 での赤枠が始点の時
（これ以降は“01”と記述する）



(d) 図 11 での青枠が始点の時
（これ以降は“11”と記述する）

図 12 今回の計算で比較する 4 通りの画像

図 12 にまとめた 4 通りの画像に対してハールウェーブレットの計算を行い，*A*・*Dl*・*Dv*・*Dd* に分割する．追加した白色部分を含めて計算した場所は，本来の画像以外の要素が含まれてしまう．そのため，*A*・*Dl*・*Dv*・*Dd* の上側・左側の 1pixel と下側・右側を“元画像の半分(切り捨て)-2”のサイズで切り取ると，白色部分を含めて計算した場所を切り取ることができる．図 12 のそれぞれの画像に対して，ハールウエーブレットの式(3)を適用した後，

白色部分を切り取った後の画像を図 13，図 15，図 17 にまとめる．

(a)“00”に対して式(3)を適用

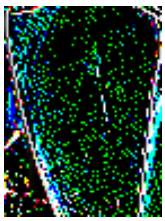
(b)“10”に対して式(3)を適用

(c)“01”に対して式(3)を適用

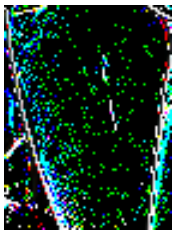
(d)“11”に対して式(3)を適用

図 13 図 12 のそれぞれの画像に対して式(3)を適用した結果の *Dl* の部分の画像

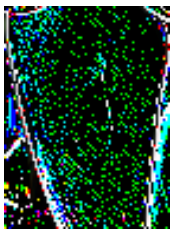
図 13(a)と図 13(c)は違いが少なく，同様に図 13(b)と図 13(d)も違いが少ない．一方，図 13(a)(b)の二つと図 13(c)(d)の二つを比べるとある程度の違いが見られる．その中でも右足部分を拡大すると違いがわかりやすい．



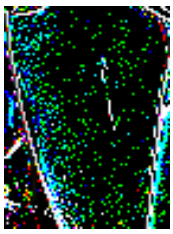
(a)“00”の拡大



(b)“10”の拡大



(c)“01”の拡大

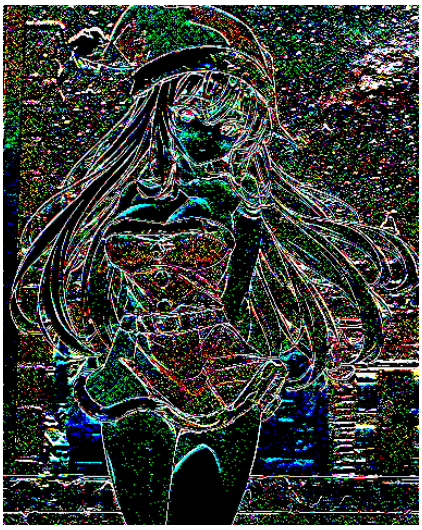


(d)“11”の拡大

図 14 図 13 の拡大部分

(a)“00”に対して式(3)を適用

(b)“10”に対して式(3)を適用

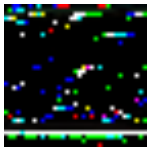


(c)“01”に対して式(3)を適用

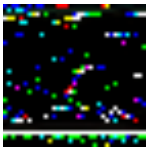
(d)“11”に対して式(3)を適用

図 15 図 12 のそれぞれの画像に対して式(3)を適用した結果の *Dv* の部分の画像

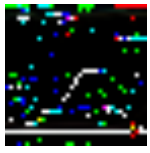
図 15(a)と図 15(b)を比較すると違いが少なく，同様に図 15(c)と図 15(d)も違いが少ない．一方，図 15(a)(b)の二つと図 15(c)(d)の二つを比べるとある程度の違いが見られる．その中でも左足の右側(適用の上あたり)を拡大するとわかりやすい．



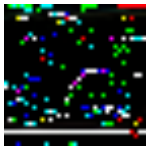
(a)“00”の拡大



(b)“10”の拡大



(c)“01”の拡大



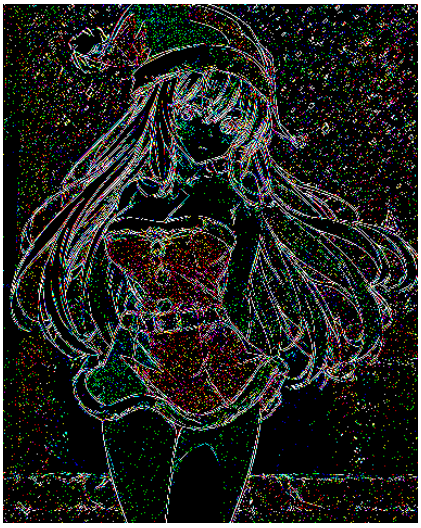
(d)“11”の拡大

図 16 図 15 の拡大部分

(a)“00”に対して式(3)を適用

(b)“10”に対して式(3)を適用

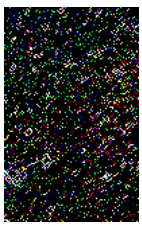
(c)“01”に対して式(3)を適用



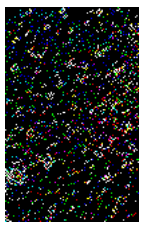
(d)“11”に対して式(3)を適用

図 17 図 12 のそれぞれの画像に対して式(3)を適用した結果の *Dd* の部分の画像

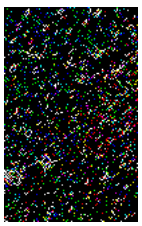
図 17 を見ると，どれもある程度異なる結果になった．これを右上部分の拡大図を用いて示したものが図 18 である．



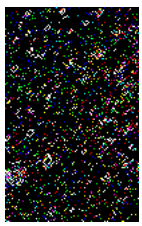
(a)“00”の拡大



(b)“10”の拡大



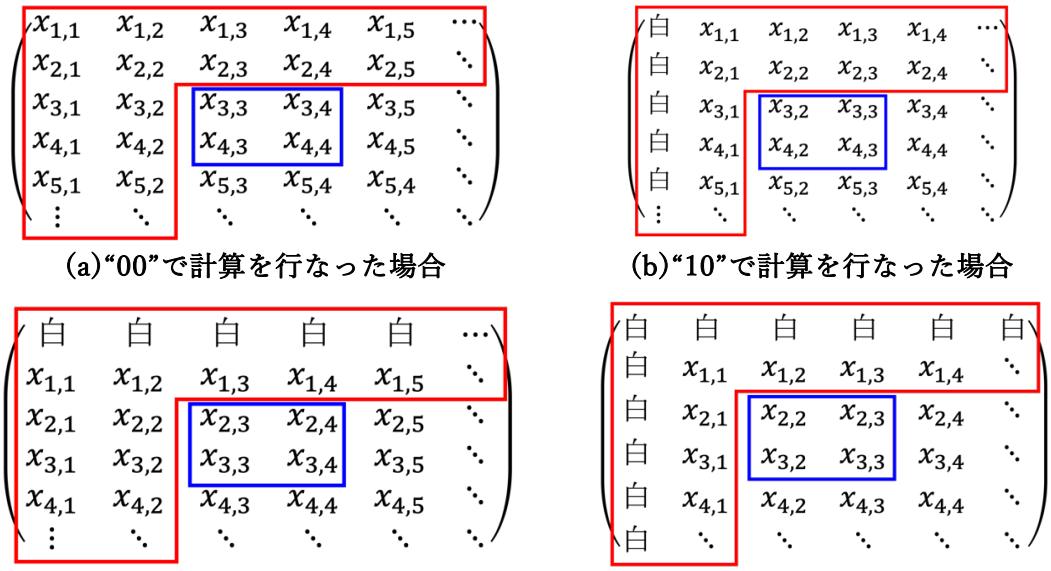
(c)“01”の拡大



(d)“11”の拡大

図 18 図 17 の拡大部分

図 13～18 の画像は結果が比較的変わっているものを選んでいるが，視覚的にはわかりにくい．図 13～18 のような画像の変化が起こる理由を，図 19 を用いて説明する．



(a)“00”で計算を行なった場合　(b)“10”で計算を行なった場合

(c)“01”で計算を行なった場合　(d)“11”で計算を行なった場合

図 19　ハールウェーブレットの計算の開始時点

図 19 の赤色で囲んだ部分は図 12 の下の文章で説明した切り取る部分であり，青色で囲んだ部分は一番初めに計算する 2×2 ブロックの部分である．*Dl* は横差分であり，求める際は(左上―右上)+(左下―右下)で計算を行なっている．そのため，左右に値が入れ替わることなく同じ部分が存在している“00”と“01”を比較すると変化が少なく，“10”と“11”も同様に変化が少なくなる．しかし，同じ部分が存在しない“00”“10”の二つと“01”“11”の二つを比較すると変化が大きくなることが多い．また，*Dv* は縦差分であり，求める際は(左上―左下)+(右上―右下)で計算を行なっている．そのため，上下に値が入れ替わることなく同じ部分が存在している“00”と“10”を比較すると変化が少なく，“01”と“11”も同様に変化が少なくなる．しかし，同じ部分が存在しない“00”“01”の二つと“10”“11”の二つを比較すると変化が大きくなることが多い．*Dd* は斜め差分であり，求める際は(左上―右上)―(左下―右下)で計算を行なっている．これは同じ部分が存在している部分が存在しないため，どれを比較しても変化が起こることが多い．

一方，人が描いたイラスト画像は図 20 のようになる．ここで，人が描いたイラスト画像には著作権があり，フリーイラストの png 画像を探しても満足がいくものを見つけることができなかったため，人が描いたイラスト画像の出力結果に似るように加工した AI 生成画像を例として用いる．例の画像を表示する際に加工しているだけで，実際の結果では人が描いたイラスト画像を用いており，図 20 のように AI 画像を加工したものは用いていない．加工としては[9]のサイトを用いて図 20 を円滑化レベル 10 で修復するのを 2 回繰り返した．図 20 の画像を用いて図 13，15，17 と同様の結果を出力したものが図 21~図 23 である．

図 20 AI 生成画像を加工し，人が描いたイラストを模擬した画像(付録 6)

(a)“00”に対して式(3)を適用

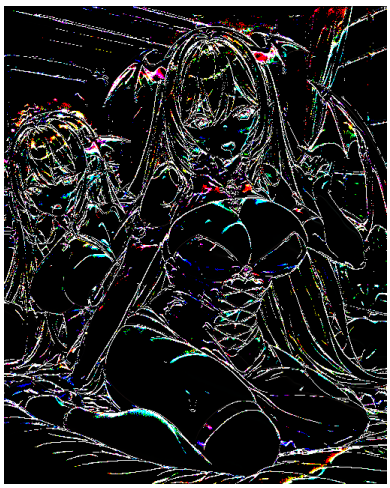
(b)“10”に対して式(3)を適用

(c)“01”に対して式(3)を適用

(d)“11”に対して式(3)を適用

図 21 図 20 の画像から“00”“10”“01”“11”の 4 通りの画像を作り，式(3)を適用した結果の Dl の部分の画像

(a)“00”に対して式(3)を適用



(b)“10”に対して式(3)を適用

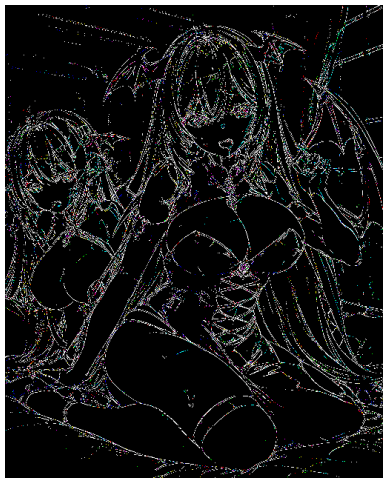
(c)“01”に対して式(3)を適用

(d)“11”に対して式(3)を適用

図 22 図 21 と同様の処理で Dv の部分の画像

(a)“00”に対して式(3)を適用

(b)“10”に対して式(3)を適用



(c)“01”に対して式(3)を適用

(d)“11”に対して式(3)を適用

図 23 図 21 と同様の処理で *Dd* の部分の画像

図 13･15･17 と図 21~図 23 を比較すると，全体的にまばらに出力されている黒色以外の色が，図 13･15･17 と比べて図 21~23 の方が少なくなっている．また，基本的に，エッジ付近や色が見てわかる程度に変化しているところにしか，白色や黒色以外の色が出力されていないことがわかる．また，図 21 の(a)(b)(c)(d)をそれぞれ比較すると，図 13 よりも画像の差が少なくなっていることがわかる．これは図 15 と図 22，図 17 と図 23 も同様の結果となる．これは，図 4 の説明で示した通り，AI 生成画像はベタ塗りやグラデーションが緩やかに変化する領域で，隣接する画素値に目に見えない程度の変化が存在することが原因だと考えられる．

3.2 JPG 画像との比較

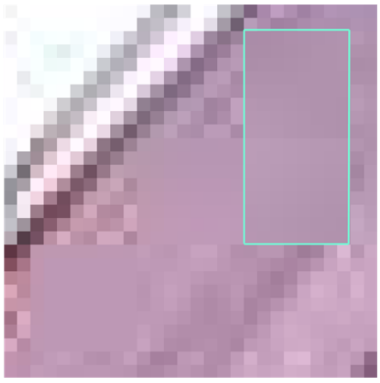
イラスト画像の拡張子には，主に png と jpg の二つが使われている．png(Portable Network Graphics)は，可逆圧縮を用いており，高品質の画像を保持することができるが，ファイルサイズが大きくなるという問題点がある．一方 jpg(Joint Photographic experts Group)は，不可逆圧縮を用いており，画像の処理や保存を繰り返すとブロックノイズやモスキートノイズなどが現れて画質が劣化するが，ファイルサイズは png に比べて小さくなる．イラストを描いて保存する際や AI を用いて画像を生成した際は png を用いることが多く，インターネット等に画像をあげる際に，容量を小さくするために jpg に変換する場合が多い．

jpg に現れるノイズであるブロックノイズやモスキートノイズを図 24 に示す．図 24 で用いる画像は，png で保存した図 9 の AI 生成画像を，python で jpg に変換する際に“quality=40”を指定して実行したものである．これにより，容量が 1．1mb→67kb に削減される．

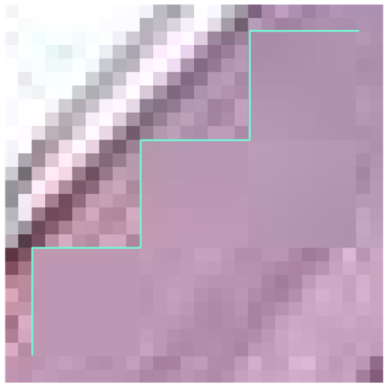
図 24 jpg 画像に発生するノイズの例

図 24 の水色の四角で囲んだところは，不規則な粒状のパターンが出力されている．これがモスキートノイズであり，画像のエッジ付近によく現れる．また jpg 画像は，画像全体を 8×8 ピクセルのブロックに分けてから，各ブロックに対して圧縮処理を行っている．そのため，拡大した右側の画像を見ると，8×8 ピクセルのブロックが出ていることがわかる．これがブロックノイズである．ブロックノイズには，grid noise，　staircase noise，corner outlier の三種類がある．それを図 25 に示す．

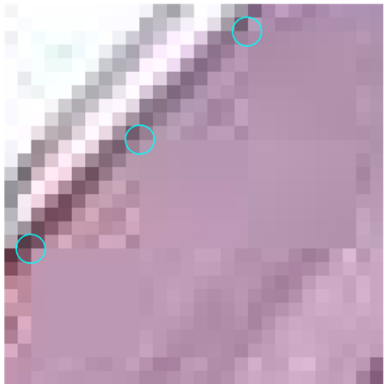
(a)元の画像



(b)grid noise



(c)staircase noise



(d)corner outlier

図 25　jpeg のブロックノイズの例

図 25(b)での四角で囲んだ部分のような，8×8 ブロック間の境界線がはっきりと見えるようなノイズを“grid noise”と呼ぶ．図 25(c)での線の部分のような，斜線が階段状に見えるようなノイズを“staircase noise”と呼ぶ．図 25(d)での丸で囲った部分のような，8×8 ブロックの角に注目すると，3 つのブロックとそれ以外の 1 つのブロックで色に差がでる現象を“corner outlier”と呼ぶ．

これらのノイズは画像を 8×8 ブロックにわけて計算を行っているため発生している．そのため，図 12 のように 4 通りに分けた後にハールウェーブレットを行うと，8×8 ブロックの境界を跨いで計算を行なってしまう．そのため，以下のように二つの画像間でハールウェーブレットの値にかなり値の差が出てしまう．ここでは python で jpg に変換する際に“quality=75”(PIL を用いた際の初期値)で png から jpg に変換したものを用いている．

(a) “00”でハールウェーブレットの計算を行なった際の *Dl*(png)

(b) “00”でハールウェーブレットの計算を行なった際の *Dl* (jpg)

(c) “10”でハールウェーブレットの計算を行なった際の *Dl* (png)

(d) “10”でハールウェーブレットの計算を行なった際の *Dl* (jpg)

図 26 png と jpg のハールウェーブレットを行なった際の比較

図 26(a)と(b)を見比べると，(b)の方が全体的に緑色が減っていることがわかる．また，

(c)と(d)を見比べると，緑色は減っているがそれ以外の色が増えていることがわかる．これは jpg 画像が 8×8 ピクセルのブロックに分けてから各ブロックに対して圧縮処理を行っているためである．これにより，png 画像で表れていた微量の色の値の変化が，圧縮処理を行うことによりエッジ付近以外の 8×8 ピクセルにおける値が削減されてしまうからだと考えられる．一方，(b)と(d)を見比べると(a)と(c)の微量の変化とは異なり，激しく色の変化が生じていることがわかる．これは，(c)の時の計算では，8×8 ピクセルのブロックを跨いでいないが，(d)の時の計算では跨いでいるからだと考えられる．このノイズは AI 生成画像だけでなく,人が描いたイラスト画像でも同様に発生する．

本研究において，隣接する画素の値の差に焦点をあてているが，jpg ではその差が jpg 圧縮の処理の影響で控えめになってしまう．また jpg から png にツールなどを用いて変換する場合も，処理の影響で隣接画素の差が控えめになってしまう．そのため，png 画像に焦点を当てて研究を行なった．

3.3 本研究で行なった比較方法

3．1．2 で記述した通り，AI 生成画像と人が描いたイラスト画像にハールウェーブレット変換を適用すると，隣接する画素値の差が強調される．この差を用いて，AI 生成画像らしさの評価値を求めることを試みた．

まず，図 13，15，17 の 12 枚と図 21~図 23 の 12 枚を例として用いる．その中でもまず図 13 の(a)(b)を例として用いる．まず対象画像の画素値の個数をそれぞれ求め，同じ画素値同士で個数の差の絶対値を求める．画素値がない場合は 0 個として計算する．それぞれの画像の画素値の個数の上位 5 個を以下に表 1 に示す．

表 1 画素値(BGR)の数の上位 5 個

[0, 0, 0] : 76272 回	[0, 0, 0] : 83770 回
[0, 1, 0] : 5796 回	[0, 255, 0]: 4639 回
[0, 255, 0]: 5630 回	[0, 0, 1] : 4630 回
[0, 0, 1] : 4602 回	[0, 1, 0] : 4444 回
[255, 0, 0]: 4065 回	[255, 0, 0]: 4015 回
図 13(a)の画素値の数の上位 5 個	**図 13(b)の画素値の数の上位 5 個**

表 1 の場合の画素値の個数の差は|76272-83770|=7498，|5796-4444|=1352，|5630-4639|=991，|4602-4630|=28，|4065-4015|=50 となる．この計算を全画素値で計算する．このように画素値の個数の差を全画素値で求め，その合計値を画像サイズ(縦×横)で割った値×100 を暫定的な評価値とする．この計算を *Dl*，*Dv*，*Dd* のそれぞれで“00”・“10”・“01”・“11”の全組み合わせ(${}_4C_2$ の 6 通りが *Dl*，*Dv*，*Dd* の 3 通りで行われるため 18 通り)で計算する．18 個の暫定的な評価値をまとめたのが表 2 である．この計算により，図 11 のようにハールウェーブレット変換の範囲を変化させたことによる画素値の変化を数値化する．

表 2 本文で説明した計算結果を図 13(AI 生成画像)と図 20(人を模擬した画像)で示す.
(小数点以下 4 桁目を四捨五入)

①“00”と“01”の比較結果(Dl)	22.287	8.331
②“00”と“10”の比較結果(Dl)	29.266	9.309
③“00”と“11”の比較結果(Dl)	29.268	9.343
④“01”と“10”の比較結果(Dl)	31.067	9.312
⑤“01”と“11”の比較結果(Dl)	31.121	9.216
⑥“10”と“11”の比較結果(Dl)	21.191	8.372
⑦“00”と“01”の比較結果(Dv)	20.308	7.222
⑧“00”と“10”の比較結果(Dv)	20.500	6.538
⑨“00”と“11”の比較結果(Dv)	22.620	7.064
⑩“01”と“10”の比較結果(Dv)	19.719	7.157
⑪“01”と“11”の比較結果(Dv)	18.715	6.713
⑫“10”と“11”の比較結果(Dv)	18.956	7.128
⑬“00”と“01”の比較結果(Dd)	11.723	3.254
⑭“00”と“10”の比較結果(Dd)	14.637	3.287
⑮“00”と“11”の比較結果(Dd)	14.197	3.128
⑯“01”と“10”の比較結果(Dd)	12.166	3.179
⑰“01”と“11”の比較結果(Dd)	12.545	3.177
⑱“10”と“11”の比較結果(Dd)	10.596	3.191
(a)なにを比較しているか	**(b)図 13 の計算結果**	**(c)図 21 の計算結果**

表 2(b)と(c)を比較すると全体的に(b)の方の値が大きくなっている．しかし，これは画像に依存し，人が描いたイラスト画像であっても(b)よりも値が大きいものも存在する．ただし，それは AI 学習対策や画像内容に遠近感を与えるために背景にノイズを入れるといったケースである．一方，表 2(c)よりも値が小さい AI 生成画像はあまり存在しなかった.

本研究では，図 11 のようにハールウェーブレット変換の範囲を変化させたことによる画素値の変化(表 2 で求めた値)を AI 生成画像と人が描いたイラスト画像の間で比較するだけでなく，ハールウェーブレット変換の範囲を変化させた 4 つの画像間での変化も求める．これを求めるために，表 2 の *Dl* の場合は①~⑥，*Dv* の場合は⑦~⑫，*Dd* の場合は⑬~⑱で 2 つずつのペアを総当たりで作り，そのペア間で表 2 の値の差の絶対値を計算する(${}_6C_2$=15 通りが Dl，Dv，Dd の 3 通りで行われる)．その結果の差をまとめたのが表 3~表 5 となる.

表 3 表 2 のそれぞれの結果の差の計算(***Dl***) （小数点以下 4 桁目を四捨五入する）

|①-②|の結果	6.978	0.978
|①-③|の結果	6.980	1.012
|①-④|の結果	8.780	0.982
|①-⑤|の結果	8.834	0.886
|①-⑥|の結果	8.077	0.971
|②-③|の結果	0.002	0.034
|②-④|の結果	1.802	0.004
|②-⑤|の結果	1.855	0.092
|②-⑥|の結果	8.075	0.937
|③-④|の結果	1.800	0.031
|③-⑤|の結果	1.853	0.127
|③-⑥|の結果	8.077	0.971
|④-⑤|の結果	0.054	0.096
|④-⑥|の結果	9.876	0.940
|⑤-⑥|の結果	9.93	0.845
(a)どこを比較しているか	(b)図 13 の計算結果	(c)図 21 の計算結果

表 4 表 2 のそれぞれの結果の差の計算(***Dv***) （小数点以下 4 桁目を四捨五入する）

|⑦-⑧|の結果	0.192	0.684
|⑦-⑨|の結果	2.312	0.158
|⑦-⑩|の結果	0.589	0.066
|⑦-⑪|の結果	1.594	0.51
|⑦-⑫|の結果	1.353	0.094
|⑧-⑨|の結果	2.121	0.526
|⑧-⑩|の結果	0.781	0.618
|⑧-⑪|の結果	1.785	0.175
|⑧-⑫|の結果	1.544	0.590
|⑨-⑩|の結果	2.901	0.092
|⑨-⑪|の結果	3.906	0.352
|⑨-⑫|の結果	3.665	0.064
|⑩-⑪|の結果	1.004	0.444
|⑩-⑫|の結果	0.763	0.028
|⑪-⑫|の結果	0.241	0.416
(a)どこを比較しているか	(b)図 13 の計算結果	(c)図 21 の計算結果

表 5 表 2 のそれぞれの結果の差の計算(*Dd*)　(小数点以下 4 桁目を四捨五入する)

|⑬-⑭|の結果	2.914	0.033
|⑬-⑮|の結果	2.474	0.126
|⑬-⑯|の結果	0.443	0.075
|⑬-⑰|の結果	0.821	0.077
|⑬-⑱|の結果	1.127	0.063
|⑭-⑮|の結果	0.439	0.159
|⑭-⑯|の結果	2.471	0.108
|⑭-⑰|の結果	2.092	0.110
|⑭-⑱|の結果	4.041	0.095
|⑮-⑯|の結果	2.032	0.051
|⑮-⑰|の結果	1.653	0.049
|⑮-⑱|の結果	3.601	0.063
|⑯-⑰|の結果	0.379	0.002
|⑯-⑱|の結果	1.570	0.012
|⑰-⑱|の結果	1.949	0.015
(a)どこを比較しているか	**(b)図 13 の計算結果**	**(c)図 21 の計算結果**

表 3~表 5 の計算結果により，ハールウェーブレット変換の範囲を変化させたことによる画素値の変化(表 2)の振れ幅がわかる．表 2 の値は，表 2 付近で説明している通り，ノイズが走った場合などでは，人が描いたイラスト画像の場合でも大きい値が出力されることがある．しかし，表 3~表 5 に示される表 2 の差分を確認することで，ハールウェーブレット変換の適用後に発生する画像の画素値の変化(この場合の変化は表 2 の値)が全体的に発生しているか，それとも局所的に発生しているかがわかる．つまり“00”“01”“10”“11”のいずれの場合でも類似した結果が出力されるか，それとも一部(例えば“00”と“10”は類似しているが，“00”と“01”，または“00”と“10”では大きく変化している)で大きな変化が生じているかを判断できる．

表 3(b)の値の最大値を図 13 の *Dl* の最終的な評価値とする．同様に表 4(b)の最大値を図 13 の *Dv* の最終的な評価値，表 5(b)の最大値を図 13 の *Dd* の最終的な評価値とする．(c)の場合も同様である．この最大値が大きいほど，“00”“01”“10”“11”のいずれかの場合で大きな変化が生じているということになる．そのため，図 21 のように，“00”“01”“10”“11”のすべての場合で似たような結果が出力されていると評価値が小さくなり，人が描いたイラスト画像の確率が高くなる．一方，図 13 のように“00”“01”“10”“11”の一部の場合で大きな変化が生じている場合，評価値が大きくなり AI 生成画像の確率が高くなる．

3.4 本研究で用いる画像について

本研究で用いる画像について説明する．人が描いたイラスト画像は著作権の都合で論文

には記載しない．画像サイズは，最小が 727×800pixel・最大が 6410×4341pixel となっており，縦長の画像も横長の画像も用意している．また，作者はすべて別人となっており，キャラクターも版権キャラクターやオリジナルの女性や男性，モンスターなどばらけさせている．背景や服装，人数もバラバラなものを選んでいる．

StableDiffusion の画像は，自身で生成したものである．VAE は“kl-f8-anime2 VAE”と“clearvaeSD15_v23”の二つを用いている．Checkpoint は “anyloracleanlinearmix_v10”・“AnythingXL_v50”・ “cocotifacute_v20”・“counterfeitV30_v30”・“cetesMix_Whalefall2”の五つを用いている．Sampling method や Schedule type・画像サイズ・step 数・seed は様々なものを用いている．また，本論文の付録に載せている画像は，全て実際にデータとして用いている画像である．一方，NovelAI と nijijourney の画像は有料サービスであるため，インターネットからダウンロードしたものとなっている．そのため，加工されているか，プロンプトや設定などは不明となっている．これらの AI 生成画像も人が描いたイラスト画像と同様に，キャラクターや背景・服装・人数などをばらけさせている．

第4章　　判別機能の評価

4.1 本論文の想定条件で評価を行った結果

本論文で用いる AI 生成画像の想定条件は，Stable Diffusion で生成した png のイラスト画像となっている．人が描いたイラスト画像の想定条件は，png のイラスト画像となっている．

3. 3 節で求めた最終的な評価値を，png 画像の AI 生成画像 50 枚と人が描いた画像 50 枚で求め，比較した結果を図 27 に示す．縦軸は画像の最終的な評価値，横軸は画像の番号となっており，1 番（左端）~50 番（中央）が AI 生成画像，51 番（中央）~100 番（右端）が人が描いた画像となっている．

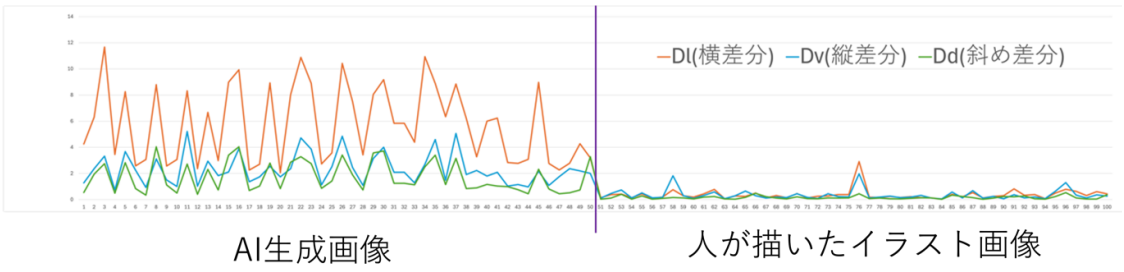
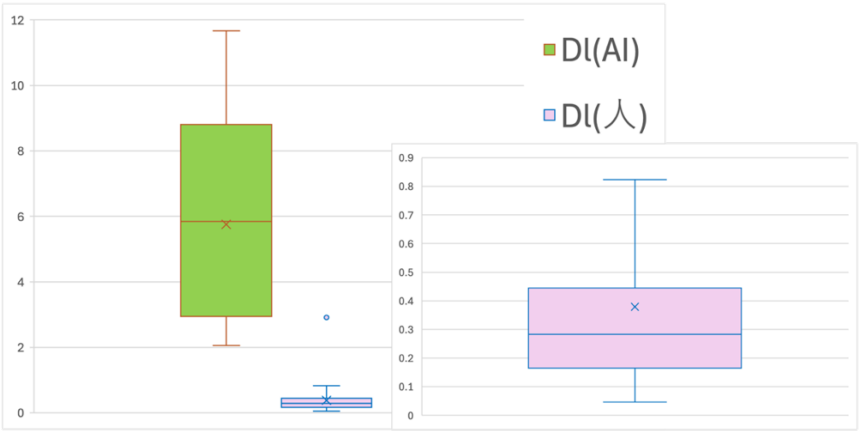
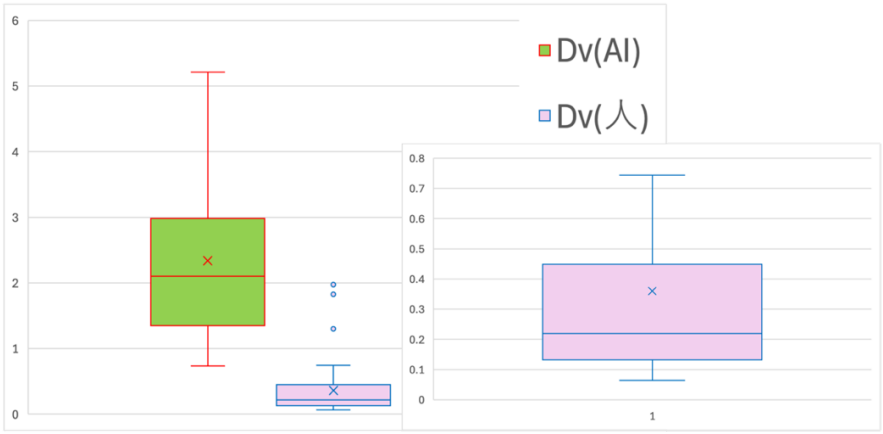


図 27 表 3~表 5 の最大値を全画像で求めたものをまとめたグラフ

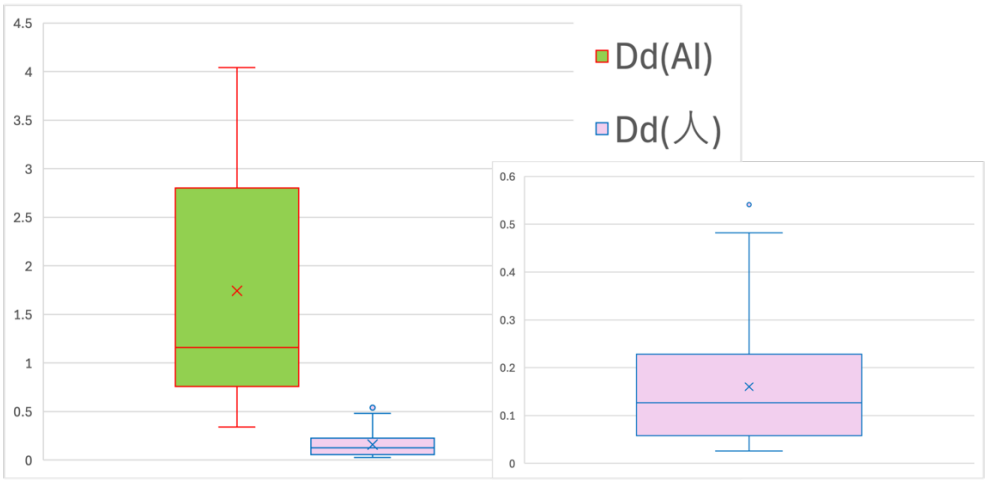
図 27 をみると，AI 生成画像の方が，全体的に値が大きくなっていることがわかる．図 27 の AI 生成画像と人が描いたイラスト画像をまとめて，それぞれの箱ひげ図を作成し図 28 に表示する．



(a)　*Dl* の箱ひげ図(右図は *Dl*(人)の外れ値を除いた拡大図)



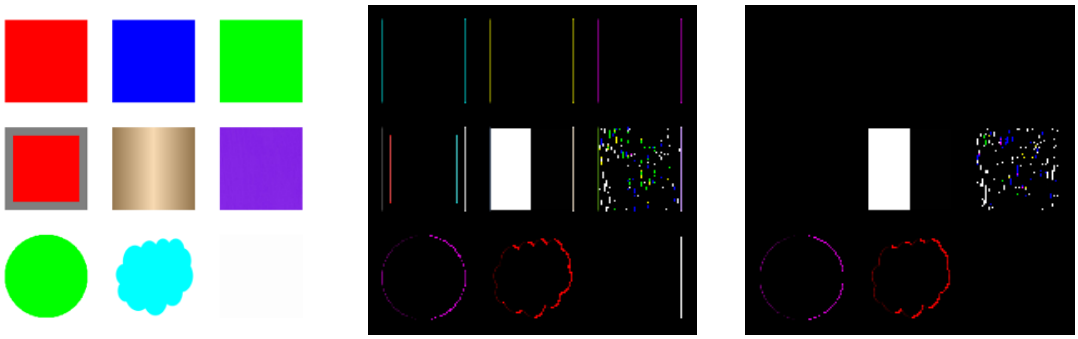
(b)　*Dv* の箱ひげ図(右図は Dv(人)の外れ値を除いた拡大図)



(c)　*Dd* の箱ひげ図(右図は Dd(人)の拡大図)

図 28 図 27 の値を箱ひげ図としてまとめたもの

図 28 の箱ひげ図は，下のひげが最小値，箱の下端が第一四分位数(25%)，箱の中にある横線が中央値(50%)，箱の上端が第三四分位数(75%)，上のひげが最大値となっている．またばつ印は平均値，丸点は excel で箱ひげ図を作成した際に外れ値として算出されたものである．この外れ値は“第三四分位数＋四分位範囲×1．5”以上のデータとなっている．図 28(a)(横差分の場合)を見ると，外れ値を除いた場合，人が描いたイラスト画像の最大値は AI 生成画像の最小値以下になっている．一方，図 28(b)(縦差分の場合)と図 28(c)(斜め差分の場合)の場合は，外れ値を除いた場合，人が描いたイラスト画像の最大値は AI 生成画像の最小値以上第一四分位数以下になっている．この結果により、*Dl* の精度はかなり高くなるが，一方で，*Dv* と *Dd* の精度は *Dl* に比べて落ちるという結果を得た．外れ値となっている画像は，机や屋根などのエッジが水平・垂直な線として描かれていたため，1pixel ずらしただけで表示され無くなったケースが多い．その例を，図 29 を用いて説明する．



(a)図 7(a)の再喝　　(b) “00”での *Dl* の結果　　(c) “10”での *Dl* の結果

図 29 図 7(a)の画像にハールウェーブレット変換を行なった結果

図 29(b)と(c)を見比べると，1pixel 動かしただけで線が表示されなくなっていることがわかる．これは，1pixel ずらしたことによりハールウェーブレットの計算がずれたことが原因である．このようなことが，外れ値となった画像で生じていることが多い．

Dv の精度が落ちる理由として，用意した画像の横方向の直線が多かったことが原因の一つだと考えられる．この影響を軽減するためには，画像数を増やすことが挙げられる．実際に画像を 20 枚→50 枚に増やした際に精度は向上した．一方 *Dd* の精度が落ちる理由として，*Dl* と *Dv* に比べて計算範囲をずらした 4 通りの差が出にくかったためだと考えられる．*Dd* も画像を増やすと精度は向上したが，*Dv* ほどの精度の上昇は見られなかった．

4.2 本論文の想定条件以外で評価を行った結果

4.2.1 AI 生成画像と人が描いたイラスト画像の両方を jpg 画像とする

上記の計算を jpg の画像で行なった結果を図 30 に示す．この場合，AI 生成画像 15 枚と人が描いた画像 15 枚について計算を行い，それぞれは右に行くほど(画像の縦×横)÷画像ファイルサイズが大きくなっている(この値が大きいほど画像の圧縮率が高くなり，品質が低くなる)．縦軸は画像の評価値，横軸は画像の番号となっており，1 番（左端）~15 番（中央）が AI 生成画像，16 番（中央）~30 番（右端）が人が描いた画像となっている．

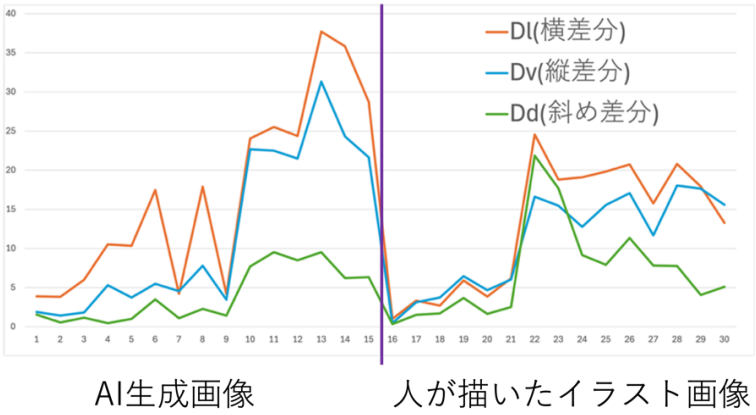
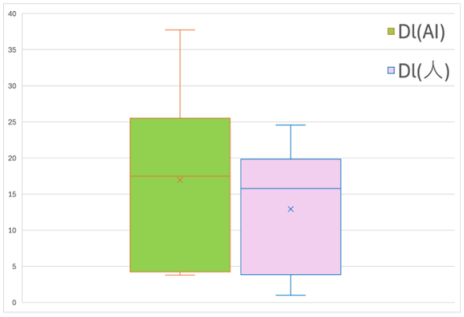


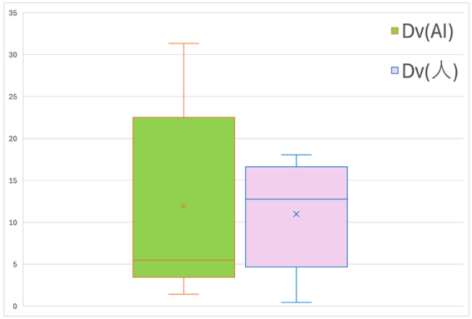
図 30　評価値を jpg 画像で求めた場合

図 30 を見ると，AI 生成画像と人が描いたイラスト画像で図 27 ほどの大きな差が生じて

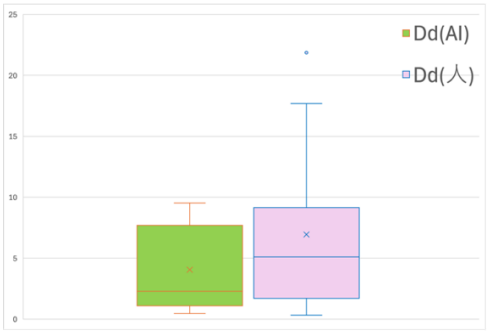
いないことがわかる．また，全体的に 1 付近よりも 15 付近，16 付近よりも 30 付近の方が値が大きくなっている．これにより，画像の品質が低くなるほど値が大きくなる傾向があると考えられる．図 30 の AI 生成画像と人が描いたイラスト画像をまとめて，それぞれの箱ひげ図を作成し図 31 に表示する．



(a)*Dl* の箱ひげ図



(a)*Dv* の箱ひげ図



(a)*Dd* の箱ひげ図

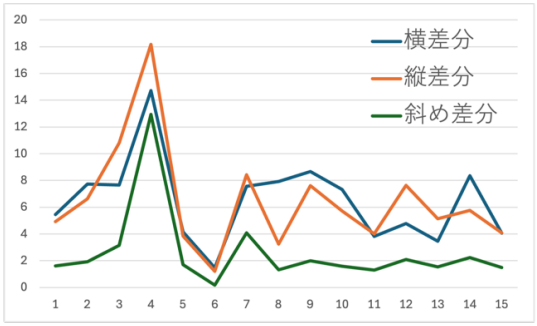
図 31 図 30 の値を箱ひげ図としてまとめたもの

図 31 の箱ひげ図を見ると，*Dl*・*Dv*・*Dd* のいずれも AI と人に大きな差がないことがわかる．これは，3．2 で説明した jpg ノイズが原因となっていると考えられる．そのため，png の場合では，値が大きいと AI 生成画像の可能性が高いと判断できたが，jpg では判断できない．

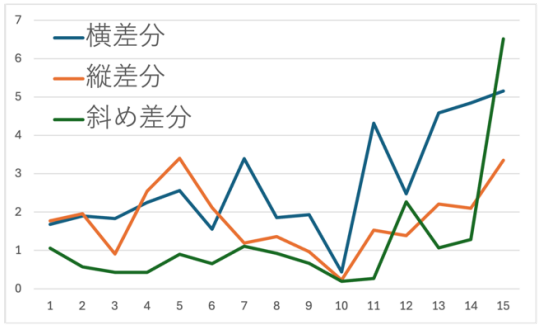
このため，jpg 画像の場合はこの方法では判別することが不可能に近いといえる．

4.2.2 別の画像生成 AI で生成した png 画像を用いる

本論文では Stable Diffusion で生成した AI 生成画像を用いて評価値を求めたが，それ以外の AI 画像生成ツールである NovelAI と Nijijourney で生成した画像を用いて評価値を求める．Nijijourney とは Midjourney を元に開発された画像生成 AI で，イラスト画像に特化している．NovelAI で生成した AI 生成画像 15 枚と Nijijourney で生成した AI 生成画像 15 枚を用いて評価値を求めたものが図 32 となっている．



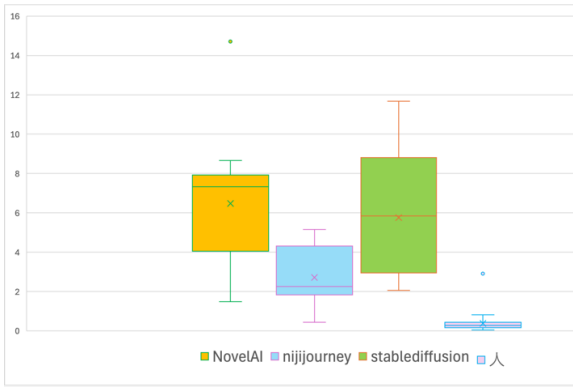
評価値を NovelAI で求めた場合



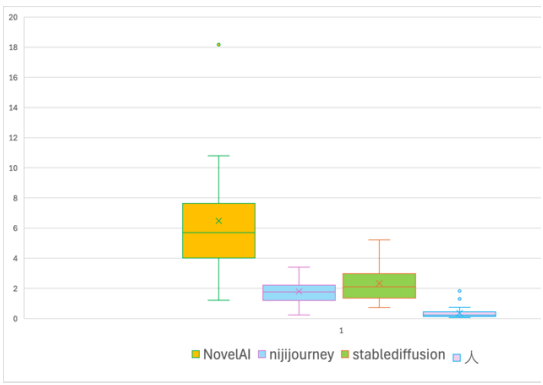
評価値を nijijourney で求めた場合

図 32 評価値を NovelAI で求めた場合

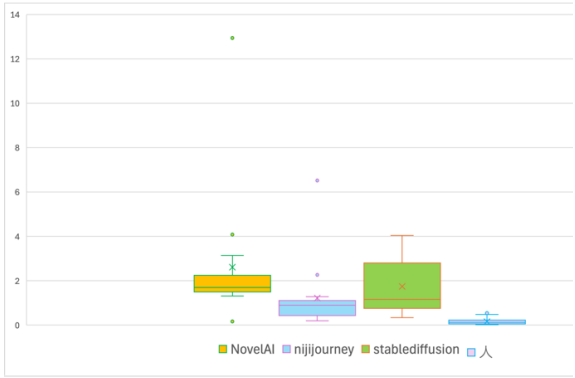
また，図 32 を差分ごとにまとめて，それぞれの箱ひげ図を作成し図 33 に表示する．比較として図 28 の箱ひげ図も載せる．



(a)*Dl* の箱ひげ図



(b)*Dv* の箱ひげ図



(c)*Dd* の箱ひげ図

図 33 図 32 と図 28 の値を箱ひげ図としてまとめたもの

図 33 の NovelAI のグラフに注目する．図 33(a)(b)(c)のすべての場合で，外れ値を除いた場合人が描いたイラスト画像の最大値は AI 生成画像の最小値以下になっている．一方 nijijourney の場合は，図 33(a)(b)(c)のすべての場合で，外れ値を除いた場合人が描いたイラスト画像の最大値は AI 生成画像の中央値以上第三四分位数以下になっている．そのため，NovelAI で生成した画像は StableDiffusion で生成した画像と同等レベルの判別が可能だと言える．一方，nijijourney で生成した画像は StableDiffusion と比べ，判別が困難になって

いると言える.

4.3 本研究で求めた評価値の仮説検定

本研究で得られた値が正規分布に従っているとは言えないため，ノンパラメトリック検定を選ぶ必要がある．そのため，仮説検定には，ウィルコクソンの順位和検定を用いる．今回は，帰無仮説(H_0)を「AI 生成画像と人が描いたイラスト画像の評価値の全体的な分布に違いはない(差は偶然)」，対立仮説(H_1)を「AI 生成画像と人が描いたイラスト画像の評価値の全体的な分布に違いがある(差は有意)」とし，有意水準 5%で両側検定を行う．data1 を AI 生成画像，data2 を人が描いたイラスト画像としている．

表 6 ウィルコクソンの順位和検定の結果(小数点以下 4 桁目を四捨五入する)・赤背景は帰無仮説を棄却，青背景は帰無仮説を棄却できない

人と何を比較しているか	(a) StableDiffusion	(b) NovelAI	(c) Nijijourney	(d)jpg 画像での StableDiffusion
統計量(Dl)	8.535	5.823	5.496	0.975
p 値(Dl)	1.407×10^{-17}	5.774×10^{-9}	3.881×10^{-8}	0.33
統計量(Dv)	8.252	5.792	5.169	0.104
p 値(Dv)	1.559×10^{-16}	6.954×10^{-9}	2.351×10^{-7}	0.917
統計量(Dd)	8.5	5.574	5.372	-1.514
p 値(Dd)	1.895×10^{-17}	2.489×10^{-8}	7.802×10^{-8}	0.13

p 値が 0.05 より小さい(赤色背景部分)と帰無仮説を棄却することができ，AI 生成画像と人が描いたイラスト画像の評価値の全体的な分布に違いがあると言える．一方 p 値が 0.05 より大きい(青色背景部分) と帰無仮説を棄却することができないため，AI 生成画像と人が描いたイラスト画像の全体的な評価値の分布に違いがないと言える．また，統計量はデータの大小を示す指標であり，大きいほど data1 の方が大きく，小さいほど data1 の方が小さい．

表 6 を見ると，png 画像で行った(a)(b)(c)はどの差分でも p 値が 0.05 より圧倒的に小さいことがわかる．そのため StableDiffusion と NovelAI と Nijijourney の画像生成 AI を使った場合，AI 生成画像と人が描いたイラスト画像のデータの分布に違いがあると言える．一方，jpg 画像で行った(d)は p 値が 0.05 より大きいことがわかる．そのため帰無仮説を棄却できないため．分布の違いに十分な証拠が得られなかった．このことから，jpg 画像は判別ができないという結果となった．また，統計量は(d)の斜め差分以外プラスの値であるため，AI 生成画像のデータの方が大きいということがわかる．

しかし，本研究では AI 生成画像か人が描いたイラスト画像かを評価値を元に比較したい．図 33 の箱ひげ図を見ると，AI 生成画像の最小値と人が描いたイラスト画像の最大値は近い値になっている．そのため，AI 生成画像で得た最小値付近の値と人が描いたイラスト画像の最大値付近で比較して有意差がでると，今回の結果で AI 生成画像か人が描いたイラス

ト画像かを評価値を元に比較できていると言える．jpg 画像は表 6 の結果より，分布に違いがないという結果を得たため省く．(a)(b)(c)の場合で AI 生成画像の最小値以上~第一四分位数と人が描いたイラスト画像の第三四分位数~最大値でウィルコクソンの順位和検定を行う．今回は，帰無仮説(H_0)を「AI 生成画像の最小値以上~第一四分位数の評価値と人が描いたイラスト画像の第三四分位数~最大値の評価値に分布の違いはない(つまり AI 生成画像と人が描いたイラスト画像を判別するのには不十分である)」，対立仮説(H_1)を「AI 生成画像の最小値以上~第一四分位数の評価値と人が描いたイラスト画像の第三四分位数~最大値の評価値に分布の違いがある(つまり AI 生成画像と人が描いたイラスト画像を判別するのに十分である)」とし，有意水準 5%で片側検定を行う．片側検定である理由は，人が描いたイラスト画像の値の方が小さい必要があるからである．

表 7 ウィルコクソンの順位和検定の結果(小数点以下 4 桁目を四捨五入する)・赤背景は帰無仮説を棄却，青背景は帰無仮説を棄却できない

人と何を比較しているか	(a) StableDiffusion	(b) NovelAI	(c) Nijijourney
統計量(Dl)	3.4641	2.7892	1.0914
p 値(Dl)	0.0003	0.0026	0.1375
統計量(Dv)	2.0207	2.5466	0.3638
p 値(Dv)	0.0217	0.0054	0.358
統計量(Dd)	3.1754	1.4552	-0.2425
p 値(Dd)	0.0007	0.0728	0.4042

表 7 において, p 値が 0.05 より小さい(赤色背景部分)と帰無仮説を棄却することができ，AI 生成画像と人が描いたイラスト画像の評価値の分布に違いがある(つまり AI 生成画像と人が描いたイラスト画像が十分に判別できる)と言える．一方 p 値が 0.05 より大きい(青色背景部分）と帰無仮説を棄却することができないため，AI 生成画像と人が描いたイラスト画像の評価値の分布に違いがない(つまり AI 生成画像と人が描いたイラスト画像が十分に判別することができない)と言える．

表 7 を見ると，StableDiffusion の場合はすべての差分で帰無仮説を棄却することができる．そのため，本研究の評価値では，StableDiffusion で生成した AI 生成画像と人が描いたイラスト画像を判別するための精度は十分であるという結果を得た．しかし，縦差分は 0.05 に近い値となっているため，少し精度が落ちる．NovelAI の場合は横差分と縦差分で帰無仮説を棄却することができ，斜め差分では帰無仮説を棄却できない．そのため，本研究の評価値では NovelAI で生成した AI 生成画像と人が描いたイラスト画像を判別するための精度は横差分と縦差分の場合は十分であるが．斜め差分では不十分であるという結果を得た．nijijourney の場合はすべての差分で帰無仮説を棄却することができない．そのため，本研究の評価値では，nijijourney で生成した AI 生成画像と人が描いたイラスト画像を判別するための精度は不十分であるという結果を得た．また，統計量を見ると(c)の斜め差分以外プラ

スの値となっているため，AI 生成画像の評価値の方が大きいことがわかる．一方，(c)の斜め差分はマイナスとなっているため, 人が描いたイラスト画像の評価値の方が大きくなっている．

第5章　　結論

本論文では，ハールウェーブレット変換を用いて，AI 生成画像と人が描いたイラスト画像を判別する手法について研究した．ハールウェーブレット変換は 2×2 ブロック単位で計算するため，その計算範囲を縦横に 1 画素ずつずらした 4 通りのハールウェーブレット変換を画像に適用した．4 通りのハールウェーブレット変換の横差分(*Dl*)・縦差分(*Dv*)・斜め差分(*Dd*)の画素値を用いて比較することで，AI 生成画像か人が描いたイラスト画像かを判定する評価値を求めた．研究の結果，*Dl* と *Dd* の精度はかなり高くなるが，一方で，*Dv* の精度は *Dl* と *Dd* に比べて落ちることがわかった．しかし，どの場合でも統計的に有意な差は出ている．*Dv* の精度が落ちる理由として，用意した画像の横方向の直線が多かったことが原因の一つだと考えられる．この影響を軽減するためには，画像数を増やすことが挙げられる．

また，StableDiffusion 以外の画像生成 AI を用いて，同様の方法で AI 生成画像か人が描いたイラスト画像かを判定する評価値を求めた．結果として，NovelAI の場合は *Dl* と *Dv* の精度はかなり高いが，*Dd* の精度は低くなる．nijijourney の場合はどの差分でも精度が低くなることがわかった．

以上の結果を踏まえ，StableDiffusion で生成した AI 生成画像と人が描いたイラスト画像をハールウェーブレット変換で判別することは可能ではあるものの，縦差分の精度は落ちるということが確認できた．一方，他の画像生成 AI で生成した画像は精度が落ちるということが確認できた．

残された課題は 3 個存在する．一つ目は png 画像にしか対応していないという点である．インターネットに上がっている画像は容量などの問題から jpg が多い．しかし，本研究の方法では jpg の画像を用いて比較することは，図 31 の結果からほぼ不可能だと言える．二つ目は，意図的なノイズを加えた場合は誤認識する可能性が高くなるという点である．意図的なノイズは，遠近感を出すためや人に注目してもらうために背景に入れることがある．また，画像生成 AI の無断学習対策にノイズが入っていることもある．そういった場合において，今回の方法は精度が悪くなってしまう．本研究でも明らかな外れ値を出力した一枚の人の画像をはぶいたが，その画像は全体的にノイズがかかっていた(意図的かどうかは不明)．三つ目は，加工で簡単に誤魔化すことができるという点である．本論文でも人が描いた画像の代わりに，AI 生成画像を加工して人が描いた画像の結果に似るようにしたが，ツールを使って全体的なノイズを除去するだけで，評価値の精度がかなり悪化してしまう．

参考文献

[1] 白井暁彦&AICU media 編集部(2024) 画像生成 AI Stable Diffusion スタートガイド SB クリエイティブ

[2] 画像生成 AI 炎上・論争・被害事例まとめ –AI 画像生成・生成系 AI 問題まとめ wiki – atwiki(アットウィキ)
https://w.atwiki.jp/genai_problem/pages/38.html (閲覧日:2025 年 1 月 9 日)

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR2022

[4] Lilian Weng. "What are Diffusion Models? | Lil' Log" July 11, 2021
https://lilianweng.github.io/posts/2021-07-11-diffusion-models/ (閲覧日:2025 年 1 月 9 日)

[5] GitHub – CompVis/stable-diffusion: A latent text-to-image diffusion model
https://github.com/CompVis/stable-diffusion (閲覧日:2025 年 1 月 9 日)

[6] 画像生成 AI によって作成されたイラストの見分け方 | おいもログ
https://blog.oimo.io/2022/10/21/human-or-ai-walkthrough/ (閲覧日:2025 年 1 月 9 日)

[7] 画像生成 AI の指に関する研究 #1-なぜ AI は指が苦手なのか | とーふのかけら
https://note.com/konapieces/n/n9aff4e3be759 (閲覧日:2025 年 1 月 9 日)

[8] 藤田 景子 研究トピックス | 富山大学理学部
http://www.sci.u-toyama.ac.jp/topics/math/201404.html (閲覧日:2025 年 1 月 9 日)

[9] ノイズの多い劣化した画像を修復 – Pix Fix
https://ja.pixfix.com (閲覧日:2025 年 1 月 9 日)

対外発表

(予定)

森川 真伍, 蚊野 浩,“ ウェーブレット変換を用いた AI 生成画像と人が描いたイラスト画像の判別に関する研究”, 情報処理学会 第 87 回全国大会 2025

謝辞

本論文を作成にあたり，丁寧なご指導を受け賜りました蚊野浩教授に感謝いたします．また，副査として論文をご精読いただきました宮森教授，川村教授に感謝いたします．

付録

本論文で掲載した AI 生成画像のプロンプトや設定を以下に載せる．

表 8 下記に載せる情報のテンプレ

Stable Diffusion で作成した画像を載せる．VAE は全て“kl-f8-anime2 VAE”を使用
ポジティブプロンプトと呼ばれる，AI に書かせたいテキスト情報を載せる．
ネガティブプロンプトと呼ばれる，AI に書かせたくないテキスト情報を載せる．
その他の情報を載せる． Checkpoint:学習モデルの選択(Stable Diffusion の学習過程で作成された画像の特徴を保存したもの．これにより生成画像のスタイルを指定できる)・Sampling method:ノイズを除去する方法の選択・schedule type:ノイズをゼロにする際の曲線比率の選択・Sampling steps:ノイズ除去を行う回数・width*height:画像サイズ・Seed:乱数の値・その他の設定は初期設定のまま

付録 1


(Best quality :1.4, masterpiece:1.3), 8K, Best illustration, Super fine illustration, 1 girl, detail eyes, cute girl, medium breasts, pajamas, pajamas with buttons, button, lying, on back, open arms for viewer, tired hair,
((worst quality, low quality)), bad anatomy, normal quality, missing limbs, missing fingers, extra fingers, extra limb, bad face, text, missing arm, disconnected limbs, bad anatomy, ugly, deformed fingers,
checkpoint: 万像熔炉|Anything XL(V5.0)・Sampling method: DPM++ 2S a・schedule type: Automatic・Sampling steps: 40・width*height: 768*960・Seed : 3823382440

付録 2

(Best quality :1.4, masterpiece:1.3), 8K, Best illustration, Super fine illustration, detail eyes, (1girl, solo), cute girl, (yandere : 1.3), long hair, looking at viewer, crazy smile, empty eyes, shaded face, (blood on face : 1.4), blood on clothes, (hold one kitchen knife), white dress, expressionless, clear face

((worst quality, low quality)), bad anatomy, normal quality, missing limbs, (missing fingers : 1.3), extra fingers, deformed fingers, extra limb, bad face, text, missing arm, disconnected limbs, bad anatomy, ugly, troubled eyebrows, blush, jpeg artifacts

checkpoint: 万像熔炉|Anything XL(V5.0)・Sampling method: DPM++ 2S a・schedule type: Automatic・Sampling steps: 40・width*height: 768*960・Seed : 1839740640

付録 3

(Best quality :1.4, masterpiece:1.3), 8K, Best illustration, Super fine illustration, 1 girl, detail eyes, cute girl, golden hair, twin tails, full body, dutch angle, small breasts, (tsurime : 1.4), cheerleader, slender, eyelid pull,

((worst quality, low quality)), bad anatomy, normal quality, missing limbs, missing fingers, extra fingers, extra limb, bad face, text, missing arm, disconnected limbs, bad anatomy, ugly, deformed fingers,
checkpoint: 万像熔炉|Anything XL(V5.0)・Sampling method: DPM++ 2S a・schedule type: Automatic・Sampling steps: 20・width*height: 768*960・Seed : 2342042311

付録 4


(Best quality :1.4, masterpiece:1.3), 8K, Best illustration, Super fine illustration, (1 girls : 1.3), detail eyes, cute girl, animal girl, animal ears, animal tail
((worst quality, low quality)), bad anatomy, normal quality, missing limbs, missing fingers, extra fingers, extra limb, bad face, text, missing arm, disconnected limbs, bad anatomy, ugly, deformed fingers,
checkpoint: 万像熔炉|Anything XL(V5.0)・Sampling method: DPM++ 2S a・schedule type: Automatic・Sampling steps: 40・width*height: 800*800・Seed : 4282951528

付録 5

(Best quality :1.4, masterpiece:1.3), 8K, Best illustration, Super fine illustration, 1 girl, detail eyes, cute girl, antenna hair, white hair, full body, (arms behind back : 1.4), dutch angle, big breasts, (red Santa Claus : 1.2), wearing miniskirt, red Santa Claus hat, outside, night sky,
((worst quality, low quality)), bad anatomy, normal quality, missing limbs, ((missing fingers)), extra fingers, extra limb, bad face, text, missing arm, disconnected limbs, bad anatomy, ugly, deformed fingers, bag
checkpoint: 万像熔炉|Anything XL(V5.0)・Sampling method: DPM++ 2S a・schedule type: Automatic・Sampling steps: 40・width*height: 768*960・Seed : 339265056

付録 6

(Best quality:1.4, masterpiece:1.3), 8K, Best illustration, Super fine illustration, (1 girl : 1.3), detail eyes, cute girl, long silver hair, full body, dutch angle, big breasts, (succubus : 1.3), on the bed, evil, curvy, (drooling : 1.3), (fang:1.2), devil wing

((worst quality, low quality)), bad anatomy, normal quality, missing limbs, missing fingers, extra fingers, (missing legs:1.2), (extra legs: 1.4), extra limb, bad face, text, missing arm, disconnected limbs, (bad anatomy : 1.2), ugly, deformed fingers, watermark
checkpoint: 万像熔炉|Anything XL(V5.0)・Sampling method: DPM++ 2S a・schedule type: Automatic・Sampling steps: 40・width*height: 1024*1280・Seed : 1529220718