VAE を用いた画像圧縮と異常検知の実現,
及びエッジコンピューティングへの応用の検討

グループ名:KYUNECTAR

今村　優希[†a)]　　　川崎　大雅[†b)]

Group: KYUNECTAR

, Yuki IMAMURA[†a)], and Taiga KAWASAKI[†b)]

あらまし　ああ（大体 300 文字）

キーワード　VAE, FPGA, エッジコンピューティング

1. は じ め に

近年，無線通信技術は飛躍的に向上しており，5G 通信の普及が進んでいる．5G は従来の 4G など通信規格と異なり，「高速大容量」「低遅延」「多数同時接続」の 3 つの特徴を備えており，その中でも「低遅延」と「多数同時接続」は新たな通信環境を構築する上で重要な軸となっている [1]．従来の 4G 通信は，人が使用するスマートフォンや携帯に焦点を当てていた．しかし，5G では車両，ドローン，センサなどの IoT 機器が大量にネットワークに接続されることを前提としている．

また，今現在の IT 業界ではクラウドが主流である [4]．図 1 の左側の処理のように，エッジデバイスからの処理をクラウドで実施し，その結果をエッジデバイスに通知するという仕組みである．ただ，多くの端末から取得したデータをクラウドのみで処理を行うのはある程度限界があり，またトラヒック量が増加して，5G のメリットを享受できないという問題が発生すると考えられる．

このような課題を解決するため，エッジコンピューティングという手法が近年注目され始めている．エッジコンピューティングとは，従来はクラウドで行っていた処理の一部を，ユーザ端末（スマートフォンや IoT 機器）の近い位置である基地局やその至近に設置されているサーバなどでデータ処理を行う技術である [2] [3]．この手法を用いることでクラウドにかかる処理をエッジコンピューティングで分散することが可能で，通信のトラヒック量の削減，5G の特徴のひとつである「低遅延」を活用することも可能である．

そこで，エッジコンピューティングの実現を VAE と FPGA を用いて実現することを考えた．すでに VAE を用いて，道路上の異物を検知するシステムも作成されており，一定の成果を得ることができている [5]．したがって，今回は車載カメラから取得した画像を VAE を搭載した FPGA で処理を行い，図 1 の右側の処理の一部を作成することを目標とした．

† 九州工業大学情報工学部 情報・通信工学科 3 年
福岡県飯塚市川津 680-4
Kyushu Institute of Technology, School of Computer Science and System Engineering, Department of Computer Science and Networks
a) E-mail: imamura.yuuki475@mail.kyutech.jp
b) E-mail: kawasaki.taiga000@mail.kyutech.jp

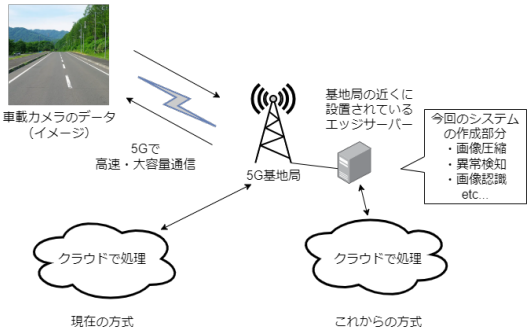


図 1: エッジコンピューティングのイメージと
今回のシステムの作成部分

表 1: 使用したツール

用途	使用ツール
VAE シミュレーション用	MATLAB 2024b
ハードウェアシミュレーション	MATLAB/Simulink 2022a
HDL コード生成	HDL Coder
FPGA 設計ソフトウェア	Vivado 2022.1
ハードウェアアクセラレーション	Vitis 2022.1
評価ボード	DIGILENT 製 ZYNQ-7010

2. 方　　法

2.1 実 行 環 境

今回のシステム作成において使用したツール及びそのバージョンを表 1 で示す．また, FPGA の評価ボードとして，DIGILENT 製の SoC を搭載した ZYNQ-7010 を使用する．

2.2 システム構造

今回の VAE を搭載したシステムでは，以下 2 つの機能を搭載した．

1　画像圧縮

VAE の特徴のひとつである次元圧縮能力を画像に応用する．

2　異常検知

もう一つの特徴である異常検知を，元画像と生成画像とを比較して行う．

全体の構想について解説する．今回使用する画像は，簡単化のためにグレースケール化したものを使用する．また，JPEG のようにブロックに分割して，それぞれのブロック毎に VAE を利用する．ブロックサイズは 16 × 16 に設定した．ブロック毎に元画像と圧縮後の画像との比較は PSNR を用いて行う．画像圧縮や異常検出はこの PSNR を用いて処理を行う．画像圧縮では，PSNR が設定した閾値以上だった場合は，圧縮した潜在空間を用いても問題がないので，潜在空間を圧縮データとして利用する．異常検知では，PSNR の値が閾値以下だった場合は道路以外の可能性が高いので異常と判定するよう設定する．

上記の機能を実現するために，VAE の構造を 2.2.1 で説明する．また，FPGA の構造の詳細を 2.2.2 にて説明し，最後に SoC FPGA の構造を 2.2.3 にて説明する．

2.2.1 VAE 構 造

VAE の構造の概略を図 2 に示す．16 × 16 の画像を使用することから，入力 256 次元，出力 256 次元で設計を行った．潜在空間の次元は，圧縮空いたとしてもある程度判別がつくように 16 次元で作成を行った．エンコーダ部分の活性化関数に関しては，平均では ReLU 関数を使用し，分散ではソフトプラス関数を使用している．また，デコーダ部分ではシグモイド関数を利用している．

$$f(x) = x \qquad : ReLU \text{ 関数} \qquad (1)$$

$$f(x) = \log(1 + e^{x}) \qquad : Softplus \text{ 関数} \qquad (2)$$

$$f(x) = \frac{1}{1 + e^{-x}} \qquad : Sigmoid \text{ 関数} \qquad (3)$$

また，VAE の学習方法を図 3 に示す．まず，道路のみのブロックを大量に用意する．そのブロックを教師データとし，VAE を学習させる．学習の条件を表 2 にまとめた．今回は道路が映った写真を用意し 3，道路のみ映った部分をブロック化することで教師データの用意を行った．また，MATLAB で VAE の学習のみ行い，そこから出力された重みやパラメータを使用して FPGA に搭載した VAE の制御を行う．

2.2.2 FPGA 構造

使用したボードは，DIGILENT 製の ZYNQ-7010 である．今回の設計では，図 4 に示すように，入力データとして，$X(16)$，$W(16\times2)$，$b(16)$，出力として $Z(16)$ を用意した（() 内は次元数）．赤色の枠で囲われているユニットは出力を $Output_1$ とすると，

$$Output_1 = X_1 \times W1_1 \qquad (4)$$

のような，入力と重みのパラメータを乗算する演算を

表 2: 学習条件

epoch	10000
eta	0.0005
Layer2(潜在空間の数)	16

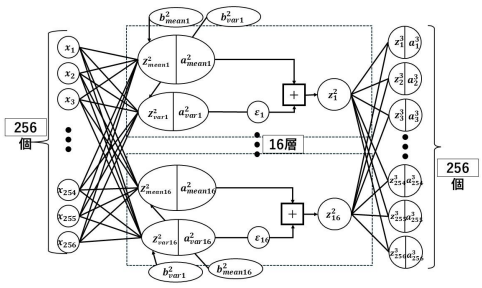


図 2: 256 × 16 × 256VAE の構造

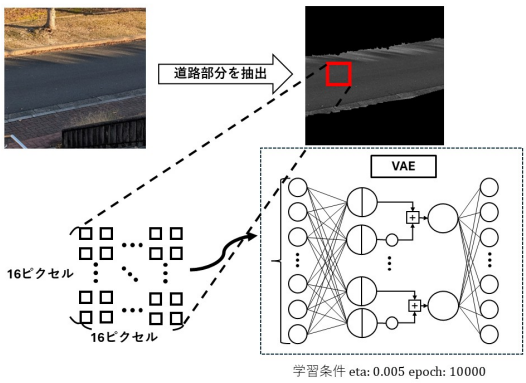


図 3: VAE 学習方法

行っている．その演算ユニットを 16 個用意したものが，青色の枠線で囲われているユニットである．最終的な一つの Z の出力は，

$$Z_1 = \sum_{i=0}^{16} X_i \times W1_i + b1 \tag{5}$$

の計算を行っている．

当初は入力 256，出力 256 の演算を FPGA に載せたかったが，容量に限界があった．したがって，今回設計した 256 × 16 × 256 の VAE を実現しやすい，X の入力が 16 になるように設計を行った．

2.2.3 SoC FPGA の構造

SoC FPGA のシステム構造の概要を図 5 にて示す．`processing_system7_0` がバスを通って様々な処理を行っおり，CPU の役割を担っている．また，`dut_forwa_ip_0` の部分が今回作成した FPGA の部分である．実装した後のリソース利用評価を表 3 に示す．

次に，SoC FPGA の処理の概要を図 6 にて示す．SD カードには，CSV ファイルに格納した重みやパラメータ，RAW 形式の画像データが格納されている．それら

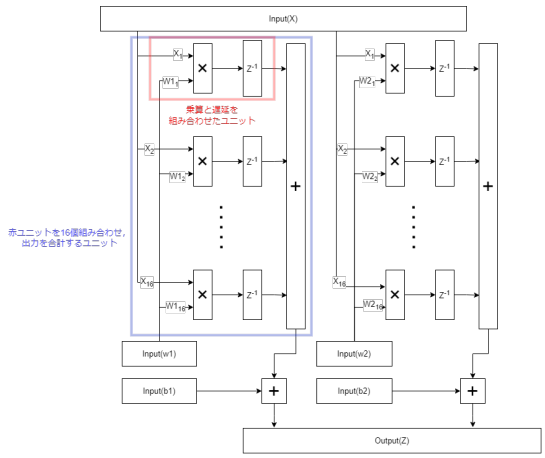


図 4: FPGA の構造

のデータを CPU が読み取る．その後，作成した FPGA に従って入力データの処理を行い，FPGA にそのデータを送信する．FPGA はデータが格納されると同時に実行し，出力結果を保存する．その出力結果を CPU が読み取りに行き，出力データを処理する．それらの処理を繰り返し行う．

今回の VAE が 256 × 16 × 256 であり，2.2.2 で作成した FPGA の構造が 16×2 である．設計した FPGA を一回稼働させることで処理できる部分を図 7 に示す．エンコーダでは，2 つの出力の計算を z^2_{mean}，z^2_{var} にあてる．一回の使用で，256 次元の入力のうち 16 次元処理できるので，1 つの潜在空間を計算させるためには FPGA を 16 回使用する必要がある．出力された z^2_{mean}，z^2_{var} を b に格納するよう設計をした．その潜在空間が 16 個あるので，エンコーダの計算は合計で 16 × 16 = 256 回稼働する．デコーダ部分を計算する際は，FPGA を一回稼働させるだけで，3 層目 Z^3_i の出力 256 次元のうち 2 次元の結果を得られる．したがって，デコーダでは 128 回 FPGA を稼働させる．このような制御を CPU で作成した．

出力で得た z^2_{mean}，z^2_{var}，z^3 は，ソフトウェア上で活性関数を用いて a^2_{mean}，a^2_{var}，a^3 の計算を行う．さらに，a^2_{mean} と a^2_{var} を用いて z^2 の計算もソフトウェア上で行っている．最後に，入力画像と出力の a^3 の値を比較し，PSNR を計測を行っている．

2.3 実験方法

今回は，図 8 に示す 3 種類の画像を用意した．図 8a は，落下物がない画像である．図 8b と図 8c は，落下物があるのをイメージしたものであり，異常検知の比

表 3: FPGA のリソース利用率

Resorce	Utilization	Available	Utilization[%]
LUT	3301	17600	18.76
LUTRAM	62	6000	1.03
FF	3297	35200	9.37
DSP	64	80	80.0
IO	12	100	12.0
BUFG	1	32	3.13

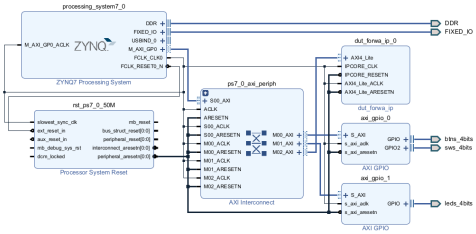


図 5: SoC FPGA の構成図

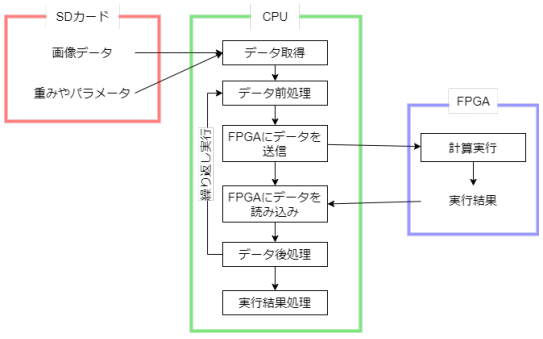


図 6: SoC FPGA の処理概要

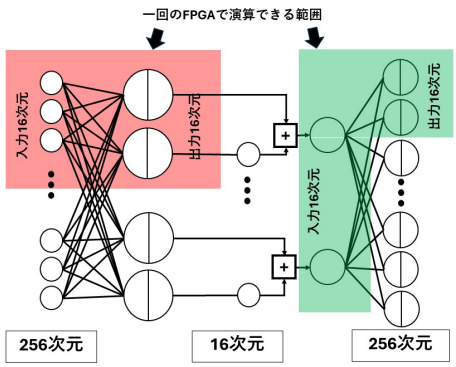


図 7: FPGA の利用イメージ

較を行うために用意した．画像のサイズは 512 × 512 で，VAE に通す際は，画像をグレー画像にしている．16×16 のブロックで分割をしながら処理を行い，PSNR を計測する．その PSNR が MATLAB の出力と FPGA

(a) もともとの画像 画像 1　**(b)** 異物設置した画像 画像 2　**(c)** 異物設置した画像 画像 3

図 8: 今回テストする評価画像

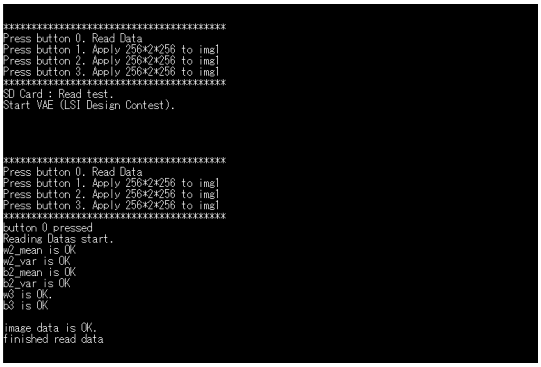


図 9: SoC FPGA の実行画面

を用いた出力でどの程度の差があるのかを比較する．FPGA で出力された PSNR は CSV ファイルに書き込み，そのデータを MATLAB 上で提示する．

3. 実験と考察

3.1 実験結果

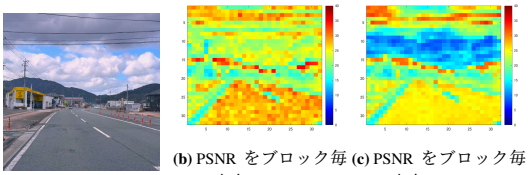
まず，SoC FPGA の実行画面を図 9 に示す．ボタン 0 を押すことで SD カードに保存されている CSV ファイルを読み込んでいる．ボタン 1 から 3 で画像を VAE に通過させる処理を行っている．図 9 はボタン 0 を押し，パラメータを取得後の状態である．

3.1.1 画像 1（図 8a）の出力比較

図 10b に MATLAB での出力，図 10c に FPGA での出力を示す．MATLAB では道路の部分付近の PSNR が 25 以上の値を示していることがわかる．加えて，空や山の一部分も PSNR が高くなっている．FPGA の出力では，MATLAB と比較して全体的に PSNR が低くなっていることがわかる．ただ，道路部分の PSNR は 25 程度と，結果としては悪くない値を出している．

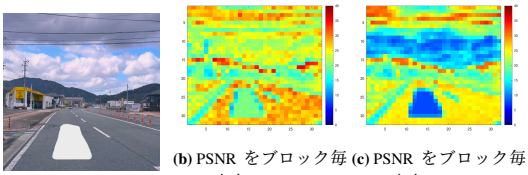
3.1.2 画像 2（図 8b）の出力比較

図 11b に MATLAB での出力，図 11c に FPGA での出力を示す．MATLAB．FPGA の出力双方とも落下物の部分の PSNR が悪化している．



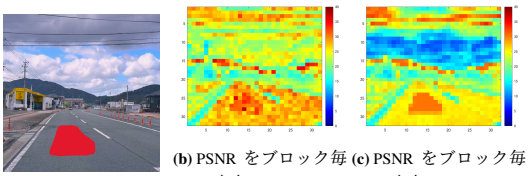
(a) テストする画像 1　(b) PSNR をブロック毎に出力 MATLAB での出力　(c) PSNR をブロック毎に出力 MATLAB での出力

図 10: 今回テストする評価画像



(a) テストする画像 2　(b) PSNR をブロック毎に出力 MATLAB での出力　(c) PSNR をブロック毎に出力 FPGA での出力

図 11: 今回テストする評価画像



(a) テストする画像 3　(b) PSNR をブロック毎に出力 MATLAB での出力　(c) PSNR をブロック毎に出力 FPGA での出力

図 12: 今回テストする評価画像

3.1.3 画像 3（図 8c）の出力比較

図 12b に MATLAB での出力，図 12c に FPGA での出力を示す．画像 2 の出力と異なり，逆に落下物がある部分の PSNR が向上している．

3.1.4 結果まとめ

それぞれの出力画像を比較して，MATLAB と FPGA の出力結果には違いが見られた．しかし，FPGA でも道路の部分が他の部分と比較して PSNR が高いため，失敗はしていないと考える．

また，異常検知に関しては，落下物の色に影響を受けることが発見された．落下物が白であると，PSNR が悪化する一方で，赤色では PSNR が向上するという結果が得られた．

3.2 考　　察

3.2.1 全　　体

得られた結果から，以下の 2 点で考察を行う．

（1） 道路以外も PSNR が高い点

（2） MATLAB と FPGA の出力が異なる点

（3） 赤色の物体も PSNR が向上している点

表 4: 画像圧縮効果

画像	ブロック数	圧縮前 [bit]	圧縮後 [bit]	圧縮率 [%]
1	346	2097152	1432832	31.68
2	305	2097152	1511552	27.92
3	349	2097152	1427072	31.95

(1) に関しては，グレー化したために生じた可能性と VAE の特性によって生じたとも考えられる．特に，空の部分はグレースケール化すると，道路の部分のように特徴量が少ない．したがって，今回作成した VAE でも表現が可能で，VAE の画像復元能力を発揮することができていると思う．

(2) に関しては，用いた変数や，固定小数点の誤差によって生じたと考えられる．ただ，詳細の出力を確認できていないため，今後検証を行って原因究明を行っていく．

(3) に関しては，グレースケール化によって生じていると考えられる．赤色をグレースケール化した際，道路の色の濃さとほぼ同じになっていた．このことから，正しく異物を判定するためにはカラースケールで処理を行う必要があると考えられる．

3.2.2 エッジコンピューティングでの採用の評価

エッジコンピューティングとして採用できるかを評価する．

まずは，画像圧縮について評価する．今回は PSNR の閾値を 25 とし，閾値以上を圧縮可能，それ未満を圧縮不可能として判定を行う．閾値が 25 以上のブロック数，圧縮前の容量，圧縮後の容量，圧縮率を表に示す．容量の計算を (6) に示す．元画像が 0 から 255 の値を取ることから 1 ピクセル 8bit である．また，潜在空間を圧縮データを用いるため，8bit の固定小数点数を使用するとする．

$$size = B \cdot 16 \cdot 8[bit] + (32 \cdot 32 - B) \cdot 16 \cdot 16 \cdot 8[bit] \tag{6}$$

どの画像でも圧縮率が 30% 近くの値を出せている．圧縮した画像をクラウドに送信し，処理を行ってもらうことで，基地局から処理サーバーまでのトラヒック量の軽減や，クラウドでの処理速度を向上することができると考えられる．

次に，異常検知について評価する．実験結果でも示したように，異物の色によって異常検知ができるか変わる．しかし，白色では異常検知が可能であることから，処理画像をカラー画像で行えば可能であると考えられる．これから走行するであろう部分を認識対象に

行えうと，認識結果を即座に車両に通知することで事故の回避ができると考えられる．

最後に，エッジコンピューティングで重要なリアルタイム性を評価する．一つの画像を入力し，PSNR を出力するのに，今回のシステムは 2 から 3 秒程度の時間を要した．このままだと異常判定を行う前に事故が発生してしまいかねない．今回は FPGA に搭載できた VAE は一部分であり，何度も FPGA を実行するためにバスを用いたデータ転送を行わないといけない．実際に 512×512 の画像を VAE で処理を行うためには，FPGA を 393216 回動作させ，そのたびにバスを用いた送受信を行っている．ただ，より高精度な FPGA を用いることでこの問題も解決できる可能性は高い．

4. 結論と今後の展望

今回は，VAE を FPGA を上に実装し，道路画像の圧縮及び異常検知を行うシステムを構築した．その結果，FPGA を用いたエッジコンピューティングにおける画像処理が実現可能であることを示した．

本システムにはいくつかの課題があるものの，適切な改善により解決可能であると考える．したがって，将来的な産業応用の可能性も秘めていると思う．

今回のコンテストで FPGA を用いた LSI 開発を体験することができた．FPGA の将来性を感じたし，AI が日常的に使用されている現在に計算を効率よく行うために必要なデバイスであることを認知した．これからも FPGA の能力を発揮できるような開発を行っていきたいし，高位合成等の FPGA を開発するための技術も日々進化しているので，様々なことに挑戦していきたい．

謝辞　今回のコンテスト開催にあたり，主催して頂いた委員会の皆さん，協賛して頂いた企業・自治体の皆さん，本当にありがとうございました．また，研究室にも配属されていないのに，様々な支援をして頂いた先生方，ありがとうございました．

文　献

[1] 森川博之, 5G 次世代移動通信規格の可能性, 岩波書店, 東京, 2020.

[2] 田中裕也, 高橋紀之, 河村龍太郎, ”IoT 時代を拓くエッジコンピューティングの研究開発”, NTT 技法ジャーナル, vol.27, no.8, pp.59-63, 2015.

[3] 横田治樹, 尾田眞也, 小林宰, 石井大二, 伊東孝紘, 五十棲淳考, ”IoT のミッシングリンクをつなぐエッジコンピューティング技術”, NEC 技法, Vol.70, No.1, 2017.

[4] 総務省, ”令和 6 年版情報通信白書”, 2024.

[5] 山本健生, 橋本敦史, 岡本大和, ”平均画像に対する VAE 異常検知の適用による道路落下物検出”, 人工知能学会全国大会論文集, 第 35 回, 2021.

[6] ”LSI Design Contest”, http://www.lsi-contest.com/, 20250131 閲覧.