

# VAEを用いた画像圧縮・異常検知を行う回路開発 及びエッジコンピューティングへの応用の検討

九州工業大学 情報工学部  
情報・通信工学科  
今村 優希/川崎 大雅



10秒ぐらい

「VAEを用いた画像圧縮・異常検知を行う回路開発, 及びエッジコンピューティングへの応用の検討」  
という題目で  
九州工業大学 情報工学部 情報・通信工学科 3年の  
今村と川崎が発表させていただきます.

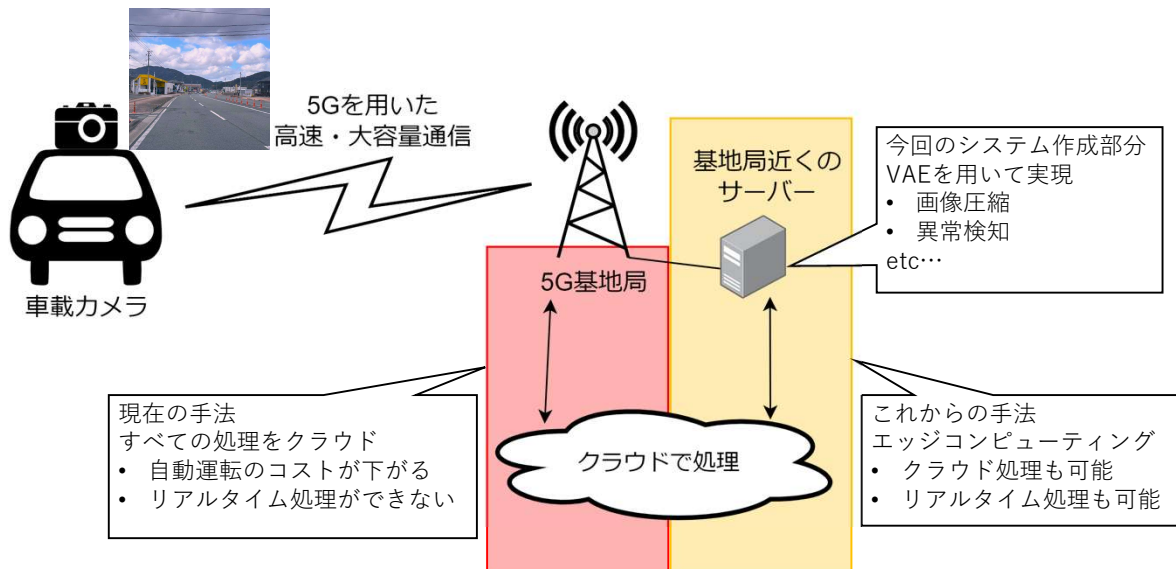
## 内容

- はじめに
- 手法
  - VAEの構造と学習方法
  - FPGAの構造と使用方法
  - SoC FPGAの構造
- 実験方法
- 実演
- 結果と考察
- エッジコンピューティングとしての活用
- 結論と今後の展望

10秒ぐらい

まずは、今回の内容です。  
前半でVAE及び作成したFPGAの構造を解説したいと思います。  
実験方法を説明した後に実演をし、結論をまとめます。

## はじめに



1分15秒ぐらい

現在、5Gが主流になってきている。

5Gは、大容量で同時多数接続、低遅延といった特徴がある。

また、4Gと異なりユーザーの端末だけでなく、自動車やドローンなどのIoT機器も接続することを目標としている。

したがって、日々使用している自動車もネットワークに繋がってくると考えられる。

そこで、今回は自動運転等に活用できるVAEの利用を考えた。

自動車がネットワークに接続され始めると、自動車の制御はクラウドが実施するようになるかもしれない。

実際にクラウドで処理を行うことで、自動運転搭載の自動車の値段が下がるといったメリットが発生する。

しかし、すべてクラウドで処理を行うとするとリアルタイム性に問題が生じてしまったり、動画像伝送に伴うトラフィック量の増加といった課題が生じる。

その問題を解決するために、エッジコンピューティングという手法が注目され始めている。

エッジコンピューティングとは、クラウドで行っている処理の一部を、エッジデバイスの近くで行う手法である。  
この手法を活用することでリアルタイム性やトラフィック量の改善が期待される。

今回はVAEを活用してこのサーバーの処理を実現することを目標として回路開発を行った。  
具体的に行う処理としては画像圧縮と異常検知である。  
また、リアルタイム性にも注意して設計を実施した。

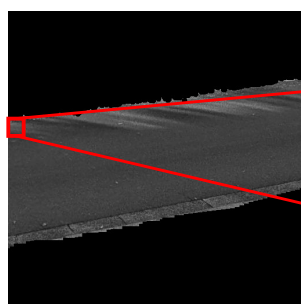
## 手法 - VAEの構造と学習方法

### VAEの概要

- 入力256次元
- 中間層16次元
- 出力256次元

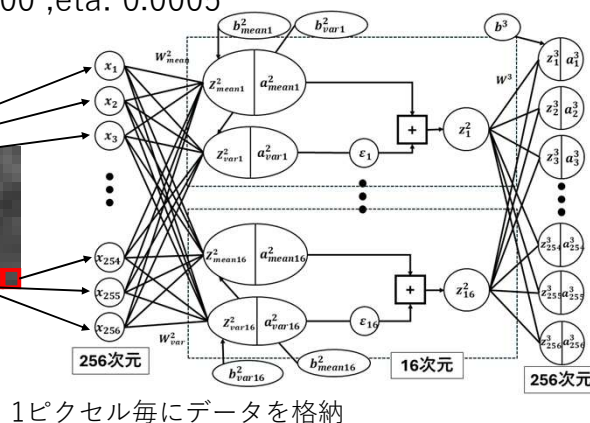
### 学習方法

- 学習として $16 \times 16$ の道路のみの画像を用意し、MATLABで実行
- epoch: 10,000 ,eta: 0.0005



道路のみを抽出

$16 \times 16$ に分割



1ピクセル毎にデータを格納

1分以内

まずはじめに設計したVAEと学習方法について解説する.

今回のVAEは入出力が256次元で中間層が16次元です.

入出力256次元の理由は、 $16 \times 16$ の画像を処理するためであり、中間層16次元は圧縮率向上と精度のトレードオフを考えた結果.

次に学習方法です.

事前に道路が写った画像を用意し、道路の部分のみを抽出する.

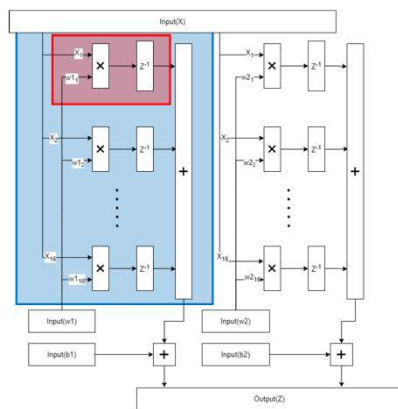
その部分のみを $16 \times 16$ のブロックに分け、そのブロックすべてを用いてVAEの学習を行わせる.

学習の条件としては、epochが一万回、etaが0.0005としている.

## 手法 - FPGAの構造及び今回の使用方法

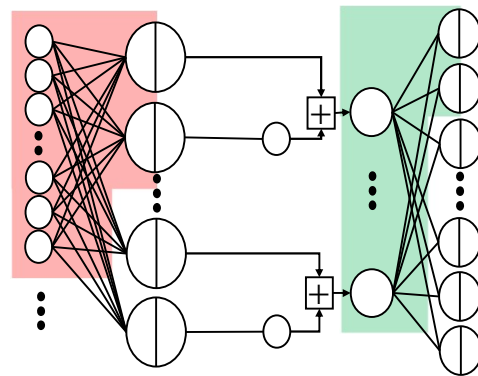
### FPGAの概要

- 入力 X: 16, w:  $16 \times 2$ , b: 2
- 出力 z: 2次元



### リソース利用率

LUT: 18.76%, FF: 9.37%, DSP: 80.0%  
エンコーダ: 256回, デコーダ: 128回



1分以内

そして、作成したVAEをFPGAに実装を行った。

FPGAの処理図は左側の図である。

入力としてXが16個、wが $16 \times 2$ 個、bが2個である。

左側の赤いユニット内でX1とw1\_1の演算を実施している。

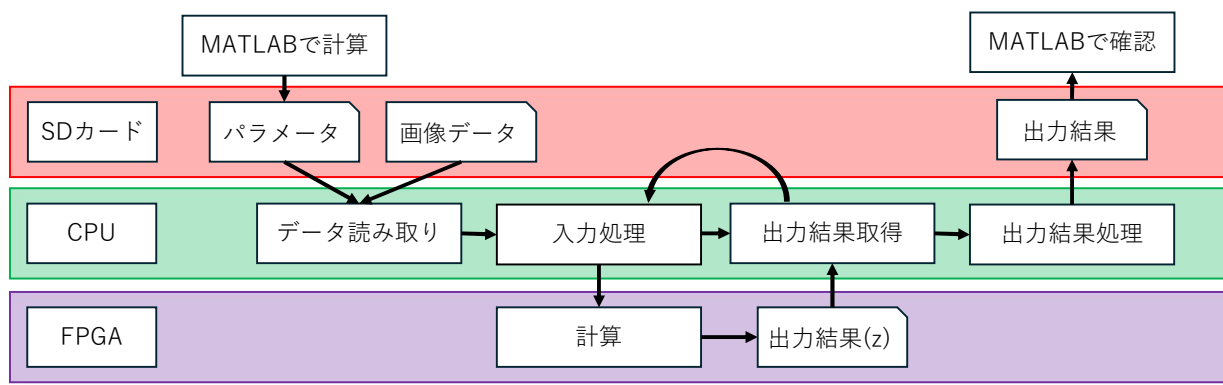
そのユニットを16個用いて、出力の合計を求めているのが青いユニットである。

実際にこのFPGAを用いて作成した入出力256次元、潜在空間16次元のVAEの計算を行わせようと思うと右側の図が示すように一部分のみしか計算できない。

エンコーダではFPGAを256回動かす必要があり、デコーダでは128回必要である。

## 手法 - SoC FPGAの構造

- 使用したSoC FPGAはZynq-7010
- VAEの学習のパラメータはMATLABで計算
  - SDカードに格納し，FPGAを用いた計算で利用する
- 出力もSDカードに保存し確認する



1分以内

最後にSoC FPGAの設計について解説する。

使用したFPGAはザイリンクスのZynq-7010である。

VAEで使用するパラメータ等は事前にMATLABで計算しておき，SDカードに保存している。

画像データもRAW形式で保存している。

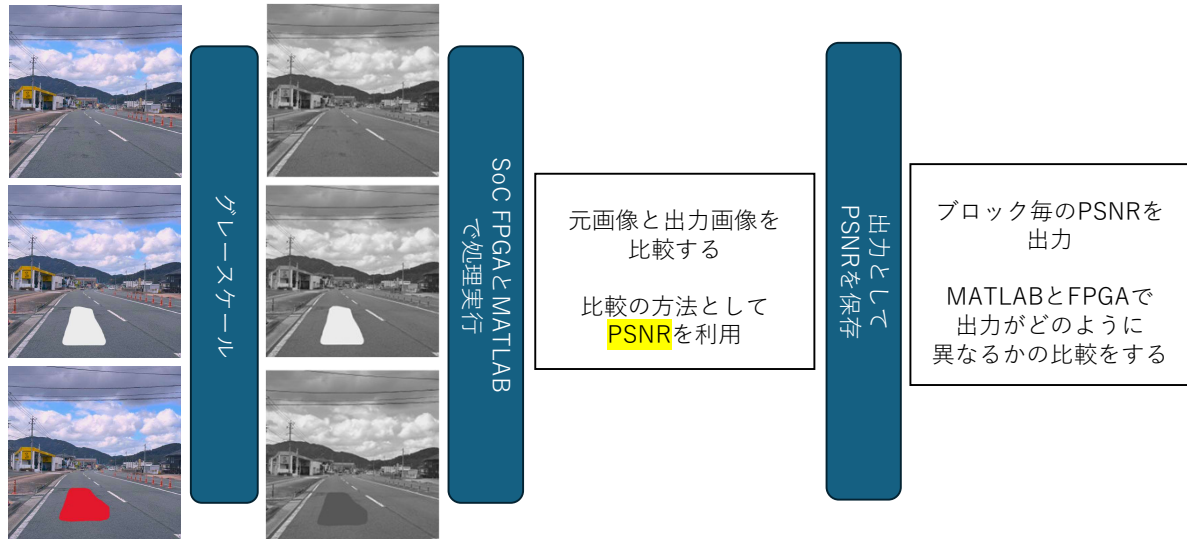
そのデータをCPUが読み込みを行っている。

CPU側で作成したFPGAに適するように入力処理を行い，FPGAはデータが格納されると同時に計算を実行する。

実行結果をCPUが取得すると，VAEが実行されるように処理を繰り返し行う。最後に出力結果をSDカードに書き出し，その結果をMATLABで確認できるように設計を行った。

## 実験方法

- MATLABとFPGAを用いて、出力結果を比較する



説明30秒 実演1分以内

SoC FPGAとMATLABそれぞれで処理を行わせる。  
今回は実験として車の中から撮影した画像を用意した。  
画像として3種類用意しており、落下物がないノーマルの画像、白色の落下物がある画像、赤色の落下物がある画像と用意した。  
それらをグレースケール化し、処理をそれぞれ行わせる。  
出力結果ともとの画像とを比較する手法としてPSNRを用いた。




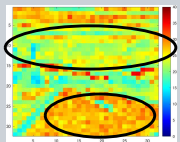
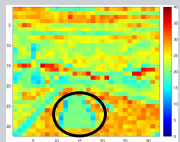
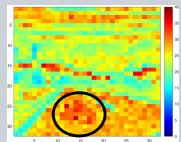
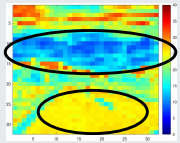
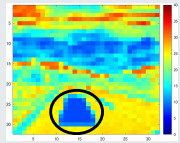
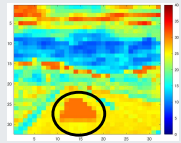
最後にブロック毎のPSNRを出力させ、出力の確認を行う。  
その際にMATLABとFPGAでどのような違いが生じているかも確認を行った。

実際にFPGAを用いた実演を行う．．．  
（もう一つのPCを用意して実施する．事前にボタンを押せば動作できる状態に持っていったく）



## 実験結果と考察

- FPGAの出力はMATLABよりもPSNRが悪化している
- 色によってPSNRが変化している

入力画像			
MATLABでのシミュレーション結果			
FPGAを用いた出力結果			

1分30秒以内

先程の出力をまとめたものを用いて結果と考察を行う。  
これらの画像は各ブロックのPSNRの出力によって色付けしたものである。  
赤に近ければ近いほどPSNRが高く、精度良く復元ができています。

画像1に関しては、MATLABの出力を見て分かる通り、道路の部分はオレンジでPSNRが高いことを示しているが、道路だけでなく雲のいち部分に関してもPSNRが高くなっている。

一方FPGAでは全体的にPSNRが低下しているのが確認できる。

ただ、道路の部分は全体的に見て良い結果が出力されているのでFPGAの構成等に大きな問題はないと考えられる。

この原因としては、パラメータの設定がうまくできていないことや、固定小数点を使用していることが原因だと考えられる。

今現在も詳細な原因を解析中である。

次に画像2と3を見比べてみると、画像2では落下物がある部分はPSNRが低下しており、正しく異常検知できるが、画像3ではPSNRが逆に向上しており、正しく異常検知を行うことができない。

これは、処理を行う前にグレースケール化を行っており、その影響で落下物部分が道路の色に近づいてしまったことから生じていると考えられる。

## エッジコンピューティングとしての活用性

- 画像圧縮

- PSNRが高いブロックは潜在空間の情報のみ伝送
- 圧縮率70%近く出ている

- 異常検知

- これから通過するであろう地点を判別させることで可能
- 落下物の色によって精度が下がる

- リアルタイム性

- 画像すべてを判別するのに4~5秒程かかる
- 課題点

解決策として

- カラーで認識できるように設計する

- 使用するFPGAを大きくする
- 高位合成を用いて効率よく設計する

1分以内

川崎担当

まず、画像圧縮について説明します。VAEを用いた圧縮では、PSNRが高いブロックは劣化が少ないため、圧縮後の潜在空間の情報のみを伝送することで、効率的なデータ転送が可能になります。また、圧縮率は約70%で十分な効果が得られました。

次に、異常検知についてです。

我々の手法では、これから通過する地点を予測する

ことが可能になりました。

しかし、グレースケール化したことで、落下物の色の違いによる影響を受け、精度が低下することがありました。

また、リアルタイム性の課題もあります。

現在の処理では4～5秒ほどかかっており、自動運転などの用途には遅すぎるという問題があります。

これらの課題を解決するために、異常検知ではカラー画像に対応させることで精度向上を目指します。

また、リアルタイム性の向上には、FPGAの規模を拡大し、高位合成を活用することで、より効率的な設計を行うことが重要だと考えています。

## 結論と今後の展望

- VAEをFPGAを用いて実装することができた
  - VAEは入出力256次元，潜在空間が16次元
  - FPGAは入力16次元，出力2次元で小さくして実現
- エッジコンピューティングとしての活用することも可能である
  - 画像圧縮率は70%程度
  - 異常検知も可能であるが改善が必要
  - リアルタイム性も改善が必要
- 5Gが普及し自動車もインターネットにつながるようになると活用されるかもしれない

1分以内

本研究では、VAEをFPGA上で実装し、画像圧縮と異常検知の可能性を検証しました。

具体的には、VAEの入出力が256次元、潜在空間が16次元、FPGAの入出力が16次元、最終的な出力が2次元での実現となりました。

エッジコンピューティングとしての活用も十分可能であり、圧縮率70%で異常検知を行えることが確認できました。しかし、実用化に向けて、カラー対応による精度向上や、処理時間の短縮が今後の課題となります。

今後、5Gが普及し自動車もインターネットにつながるようになると、活用が期待される技術の一つだと考えられます。

特に、自動運転技術においては、巨大なデータ量の処理とリアルタイムでの分析が不可欠です。

そこで、本研究のようなエッジコンピューティング技術を活用することで、従来のクラウド中心の処理手法と比べてレイテンシを最小限に抑え、自動運転の精度と処理速度を大幅に向上させることが可能になります。

これにより、安全性の向上や交通の最適化が期待され、より実用的な自動運転社会の実現に貢献できるのではないかと考えています。

## 補足資料 - VAEの学習状況

## 補足資料 - VAEの中間層が16次元の理由

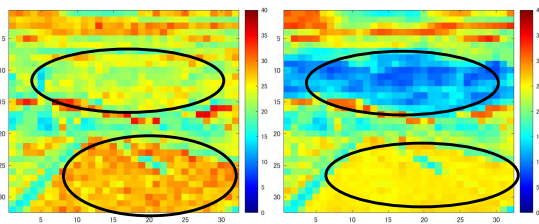
## 補足資料 - 8bitで問題ないか



## 補足資料 - 考察の詳細

FPGA出力のPSNRが全体的に悪化している点

- パラメータの設定が問題
- 固定小数点による誤差
- 道路の部分はPSNRが高い
  - 判別には問題なさそう



赤色ではPSNRが良い点

- グレースケールによる影響
- 学習データに問題がある

