

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.

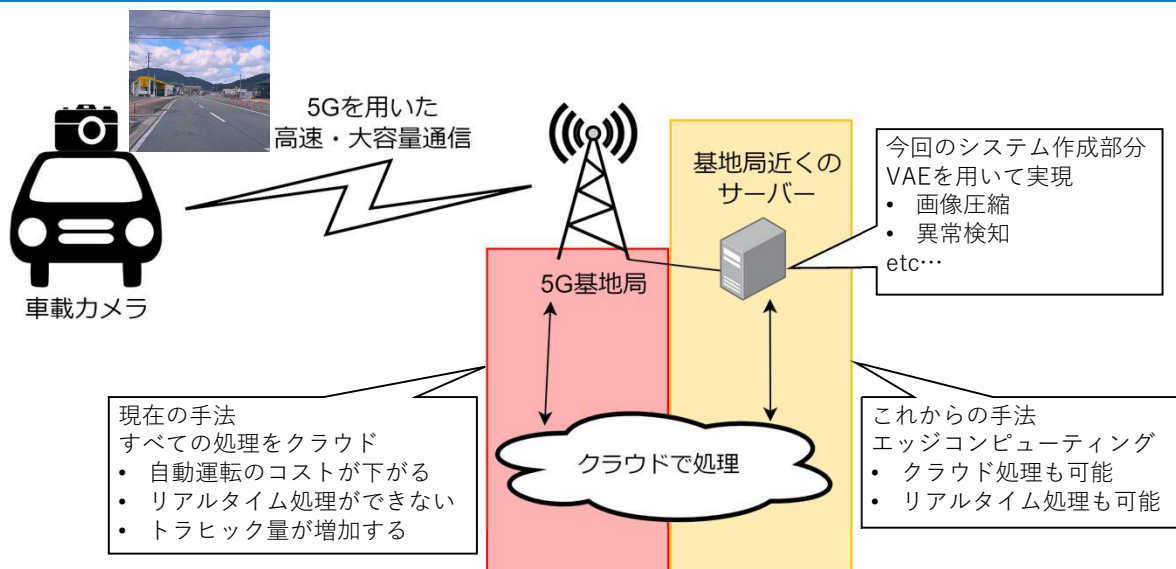
内容

- 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30秒ぐらい

現在、5Gが普及してきている。

5Gが普及するにあたり、自動車もネットワークに接続されると考えられる。すでに自動運転車が開発・普及され始めているが、自動車がネットワークに接続され、自動運転の処理をクラウドで行うことができるようになる。クラウドで実施することで自動運転車の価格が下がるというメリットがある。

しかし、すべての処理をクラウドで行うとするとリアルタイム処理ができないことやトラヒック量の増加といった問題が生じる。

それら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ヒ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行う回路開発を行った。

自動車の車載カメラから道路を撮影し、その動画像を送信するために圧縮を行う処理と、落下物がないかを判断する異常検知の処理を実現する。

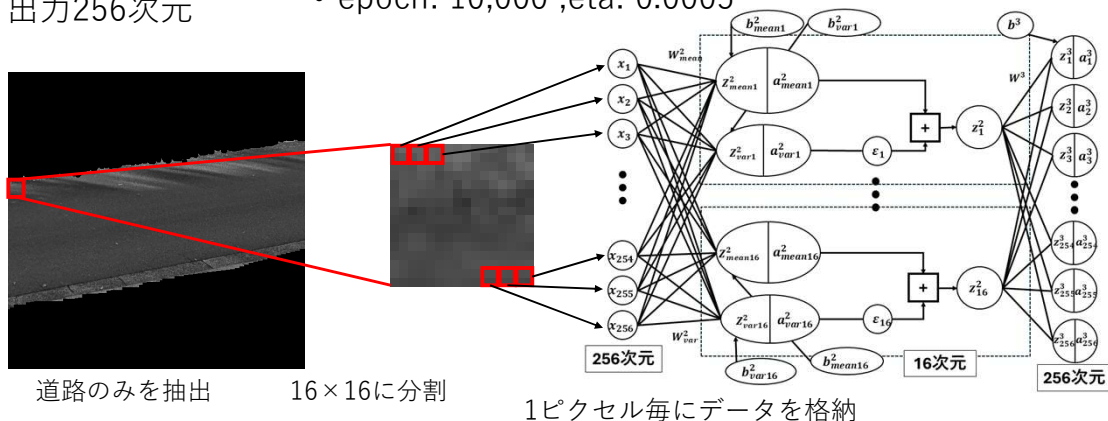
手法 - VAEの構造と学習方法

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 16×16 の道路のみの画像を用意し、MATLABで実行
- epoch: 10,000 ,eta: 0.0005



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、 16×16 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。

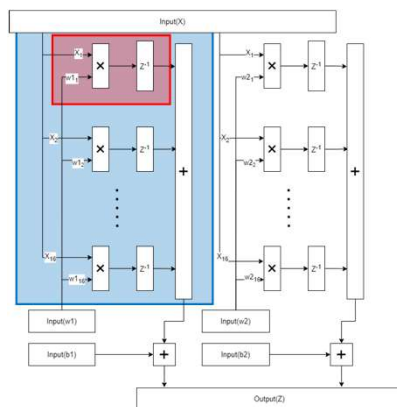
その部分のみを 16×16 のブロックに分け、そのブロックすべてを用いてVAEの学習を行わせる。

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

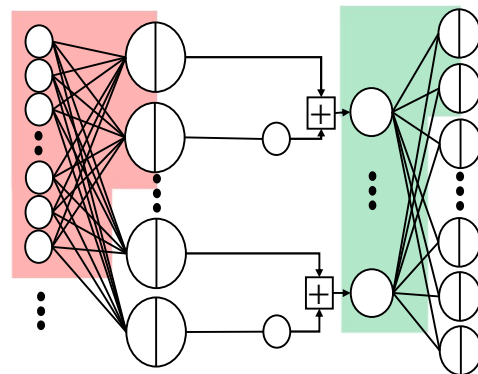
FPGAの概要

- 入力 X : 16, w : 16×2 , 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を用いて実装を行う。

FPGAの処理図は左側の図である。

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。

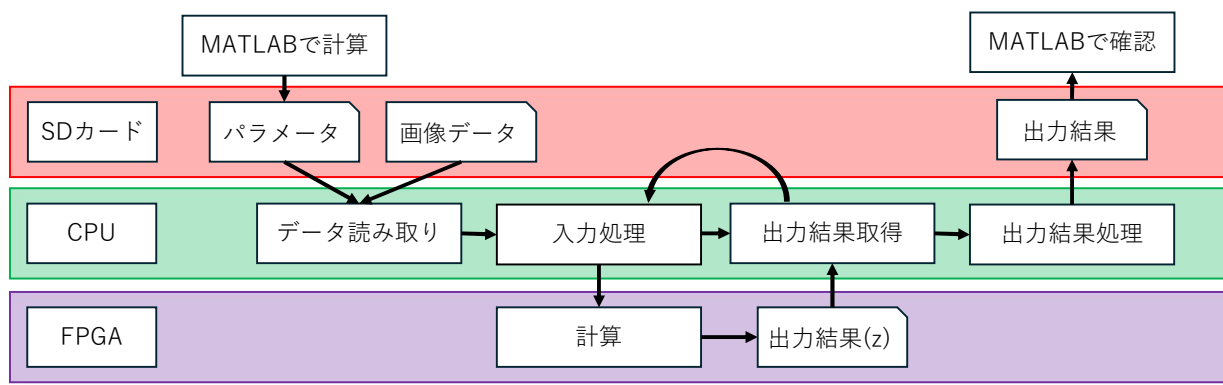
そのユニットを16個用いて、出力の合計を求めているのが青いユニットである。

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。

使用したデバイスはザイリンクスのZynq(ジンク)-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CSV形式で保存している。

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。

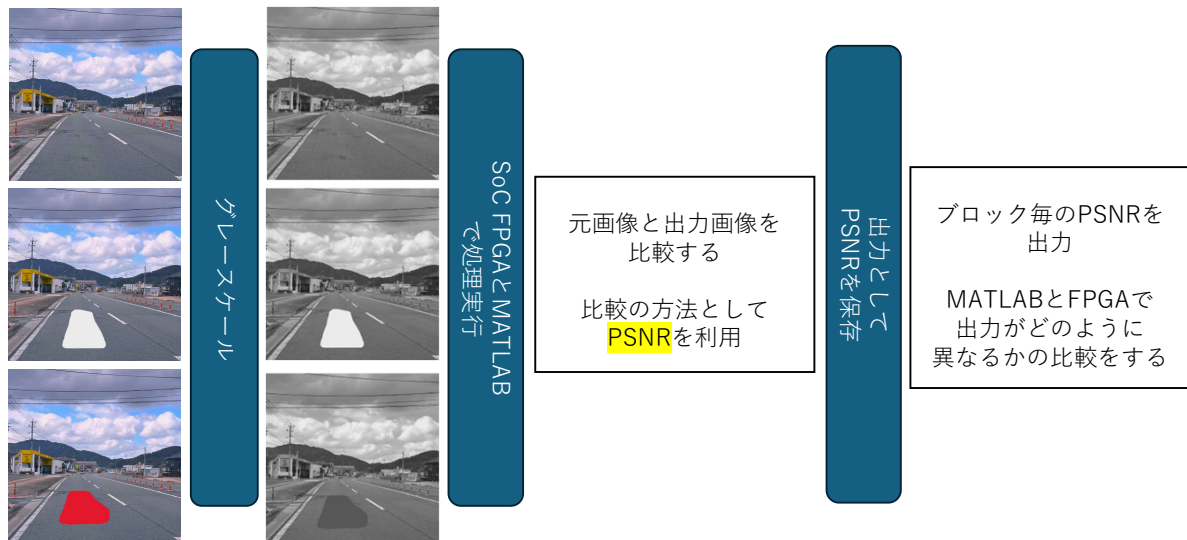
FPGAに適するようCPUが入力処理を行い、FPGAはデータが格納されると同時に計算を実行し、CPUがその処理結果を取得する。

この動作を繰り返し行うことでVAEの実現を行っている。

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

次に実験方法を解説します。

比較対象のために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。

サイズが512×512の画像を3種類用意している。

それぞれ、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を出力させ確認を行う。




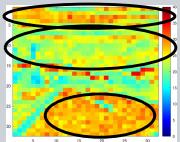
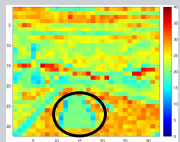
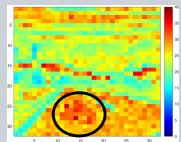
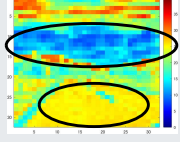
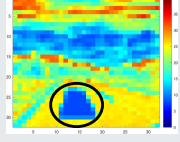
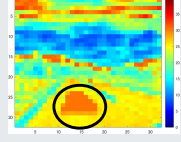
その際にMATLABとFPGAでどのような出力の違いが生じているかも確認を行った。

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る。

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。

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による誤差が生じているものだと考えられる。

今現在も詳細な原因を解析中である。

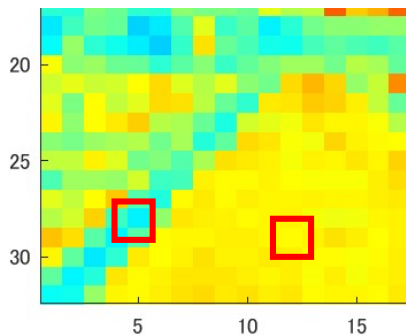
ただ、道路の部分は全体的に見て良い結果が出力されているのでFPGAの構成等に大きな問題はないと考えられる。

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。

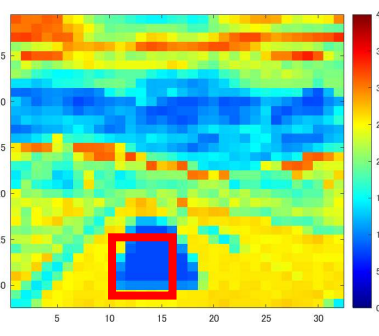
エッジコンピューティングとしての活用性

• 画像圧縮



- PSNR25[dB] 以上
 - 潜在空間の情報を8bitで送信
- PSNR 25[dB] 未満
 - 元のデータを送信
- 結果的に約70%の圧縮効果

• 異常検知



- 一部分のみで判定
 - 画像1：40.625%
 - 画像2：71.875%
 - 画像3：37.500%
- カラー画像の処理等で解決

• リアルタイム性

- 4~5秒程度の処理時間
- 解決策：
 - FPGAを大きくする
 - 高位合成等を用いた効率の良い設計

1分以内

川崎担当

次に「エッジコンピューティングとしての活用性」について説明します。
まず、画像圧縮についてです。PSNRの値が25以上のときは画質の劣化が少ないため、潜在空間の情報を8bitで送信し、25未満のときは元のデータを送信する方式にしました。その結果、全体で約70%の圧縮効果を得ることができました。

次に、異常検知についてです。特定のエリアに注目して落下物の判定を行い、その部分でPSNRの値が25未満となる割合を算出しました。
道路のみが写った画像1では40.625%、白い落下物がある画像2では71.875%、赤い落下物がある画像3では37.5%となりました。
しかし、グレースケール化の影響で落下物の色により、精度が低下する課題が明らかになったので、カラー画像に対応させることで解決できると考えています。

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。

FPGAの規模を拡大や、高位合成等を活用することで、より効率的な設計を行うことで解決できると考えています。

<Short.ver>

次に、エッジコンピューティングとしての活用性について説明します。

画像圧縮では、PSNRが25以上のときは潜在空間の情報を8bitで送信し、25未満のときは元データを送信する方式を採用し、約70%の圧縮効果を得ました。異常検知では、特定エリアのPSNR値を分析し、落下物の判定を行いました。道路のみの画像1で40.625%白い落下物がある画像2で71.875%、赤い落下物がある画像3で37.5%の割合を示しました。グレースケール化の影響で精度が低下したため、カラー対応が課題です。

また、処理時間が4～5秒と長く、自動運転技術での活用には不十分なため、FPGAの規模拡大や高位合成の活用で効率化を図ります。

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。

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した。

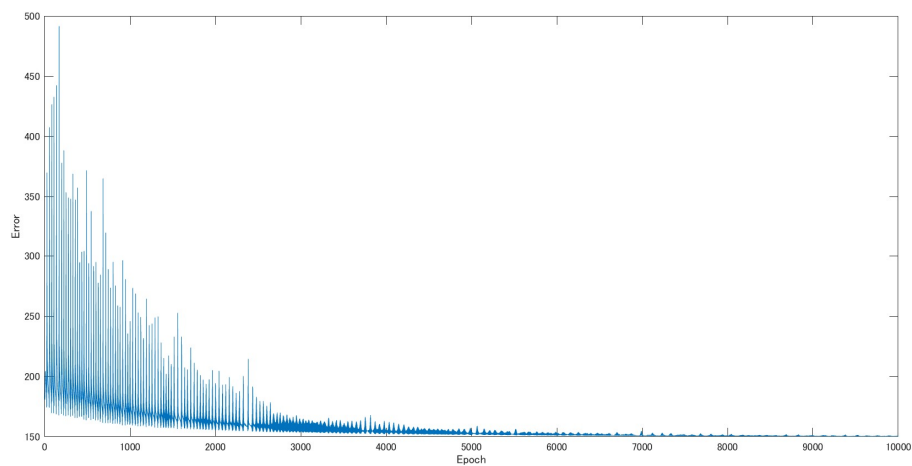
<Short.ver>

本研究では、VAEをFPGA上で実装し、画像圧縮と異常検知を検証しました。圧縮率70%で異常検知が可能であり、エッジコンピューティングとして有用であることが確認できました。今後の課題は、カラー対応による精度向上と処理時間の短縮です。

以上で今村・川崎の発表を終わります

補足資料 - VAEの学習状況

- 学習条件
 - epoch: 10,000 eta: 0.0005



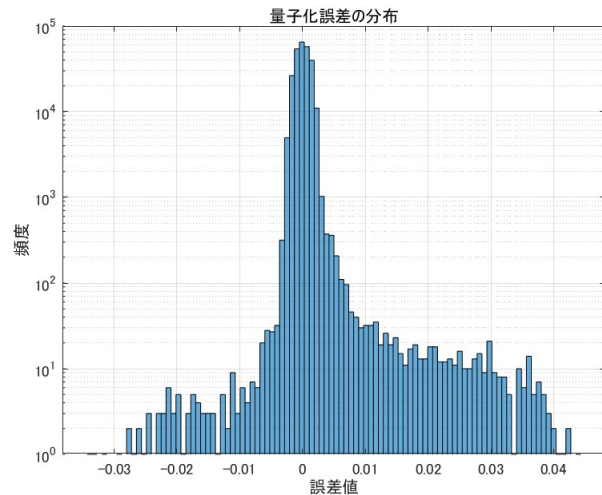
補足資料 - VAEの中間層が16次元の理由

補足資料 - 8bitで問題ないか

- 潜在空間を8bitにした後に計算を行った結果

- 潜在空間 z を
doubleのまま計算した a_3 と
8bit固定小数点で計算した a_3
の誤差等を比較

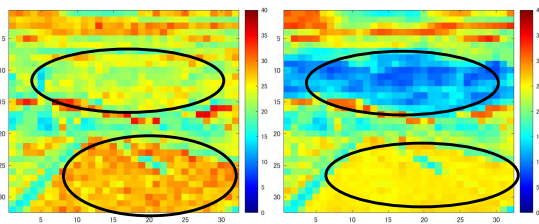
- 誤差は0付近に集まっている
- PSNRも54.88[dB]
- 出力に問題はない



補足資料 - 考察の詳細

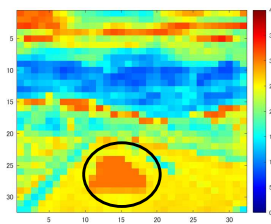
FPGA出力のPSNRが全体的に悪化している点

- 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



補足資料 - エッジコンピューティングの活用(文字)

- 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。