RealSenseカメラを用いた機械学習による深度推論の検証

平石研究室　S19096　小島 龍輝

はじめに

前期での研究内容について

　RealSenseカメラを使ったRGB画像とRGB-D画像ペアのデータセットを作成し，GitHubにて公開されていたPix2Pixライブラリを用いた推論を実行した.

後期研究の内容

- データセットの拡充
- 機械学習ネットワークの変更
- 学習した推論データの検証
  →さらなる精度向上のとりくみ

研究過程–データセット

前期では160枚の画像ペアで学習を行った.
→これに対し,追加撮影を行い456枚へと増強を行った.

入力画像として用いる画像サイズが256×256Pixel
→元画像から256×256Pixelの画像を50Pixelごとに切り出し位置をずらして多数作成.

456枚から10944枚(1 →24)のデータセットを作成した.

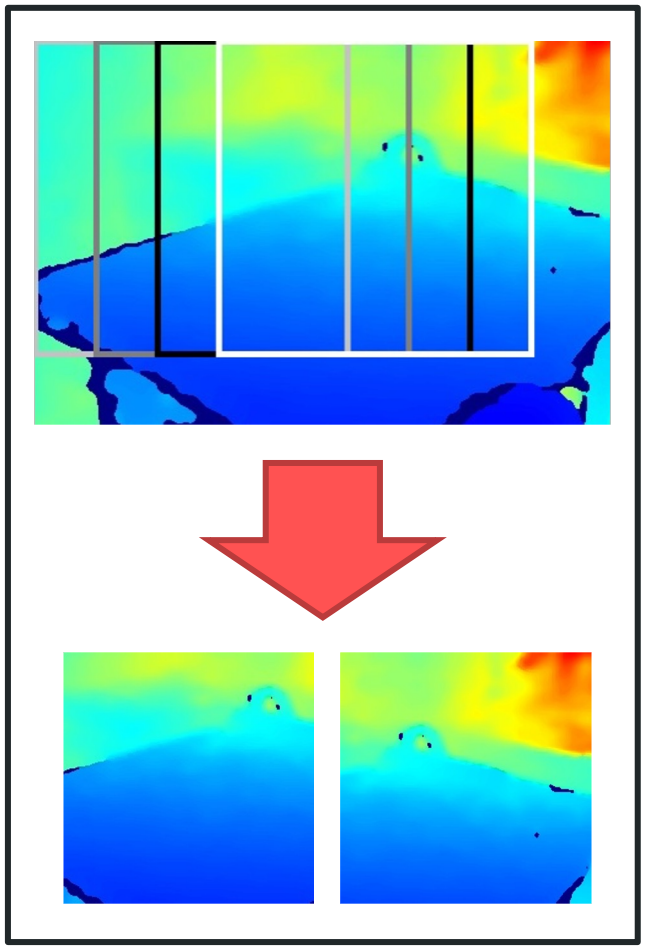

図1 データセット画像の切り出し例

研究過程–PytorchによるPix2Pix実装の経緯

前期からの課題

　TFによるPix2Pixライブラリにおいて，バージョンの問題と思われるエラーが多発し学習を継続することが困難であると判断.

Pytorchライブラリを用いてPix2Pixを実装することにした.

　参考元サイトにて公開されていたソースコードを改変し，Gray to Color対応のネットワークを，RGB to RGB-Dへの対応を行った.

　その他，ネットワーク形状を今回使用する256*256Pixelに対応させる変更を行った.

参考サイト：pix2pixを1から実装して白黒画像をカラー化してみた（PyTorch）
https://blog.shikoan.com/pytorch_pix2pix_colorization/ (2023.1.20 参照)

構築したネットワーク形状

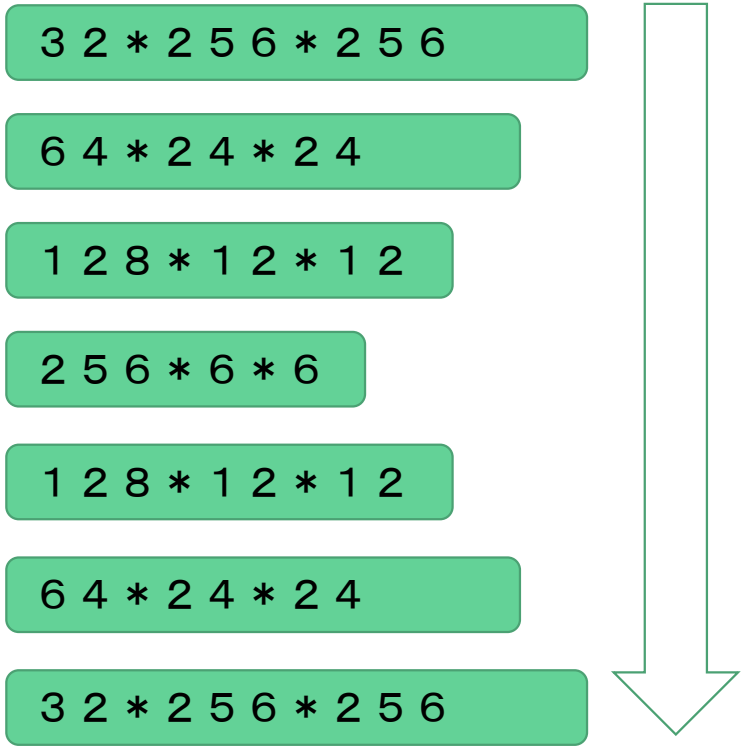

図2 ネットワーク形状

使用したネットワークは，エンコーダー・デコーダー形式のU-Net.
畳み込みのフィルタにより次元を圧縮し，活性化関数にReLUを使用している.

```python
class Generator(nn.Module):
    def __init__(self):
        super().__init__()

        self.enc1 = self.conv_bn_relu(3, 32, kernel_size=5) # 32x256x256
        self.enc2 = self.conv_bn_relu(32, 64, kernel_size=3, pool_kernel=4)  # 64x24x24
        self.enc3 = self.conv_bn_relu(64, 128, kernel_size=3, pool_kernel=2)  # 128x12x12
        self.enc4 = self.conv_bn_relu(128, 256, kernel_size=3, pool_kernel=2)  # 256x6x6

        self.dec1 = self.conv_bn_relu(256, 128, kernel_size=3, pool_kernel=-2)  # 128x12x12
        self.dec2 = self.conv_bn_relu(128 + 128, 64, kernel_size=3, pool_kernel=-2)  # 64x24x24
        self.dec3 = self.conv_bn_relu(64 + 64, 32, kernel_size=3, pool_kernel=-4)  # 32x96x96
        self.dec4 = nn.Sequential(
            nn.Conv2d(32 + 32, 3, kernel_size=5, padding=2),
            nn.Tanh()
        )
```

図3 ネットワーク形状部ソースコード

研究過程 -Pix2Pixの学習結果

学習を行った結果について

→学習の進行に応じて生成器（Generator）の損失が増加して減少するという山があるものの，山の後の損失の差が少ないため収束したと判断した.

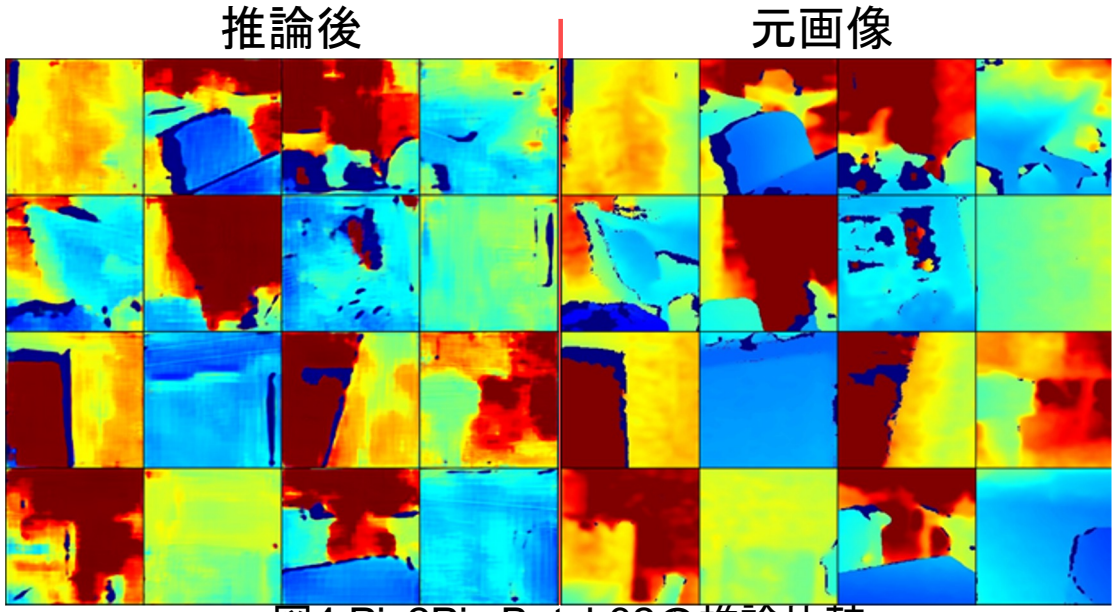

図4 Pix2Pix Batch32の推論比較

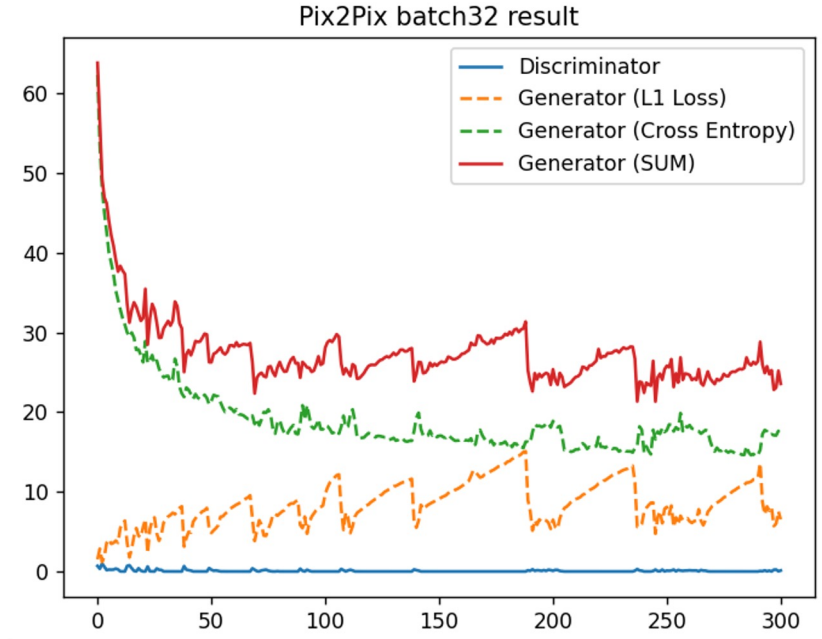

図5 Pix2Pix Batch32の学習曲線

研究過程 -混合精度手法について

上記のPix2Pixネットワークでは，VRAMの関係から，Google Colabなどの環境ではこれ以上ミニバッチサイズを拡充することが不可能であった。

→混合精度手法を使用.

混合精度（Mixed Presicion)とは

学習の際に，FP32（単精度浮動小数点数）ではなくFP16（半精度浮動小数点数）を変わりに使用することで，学習に必要なメモリを半減させる手法.

研究過程 -混合精度手法の学習結果

学習を行った結果について

Batch32と比較すると収束がなだらかで、300エポック以降も学習が進行した.
550～600エポック点にて急速な悪化が見られたため学習を停止.

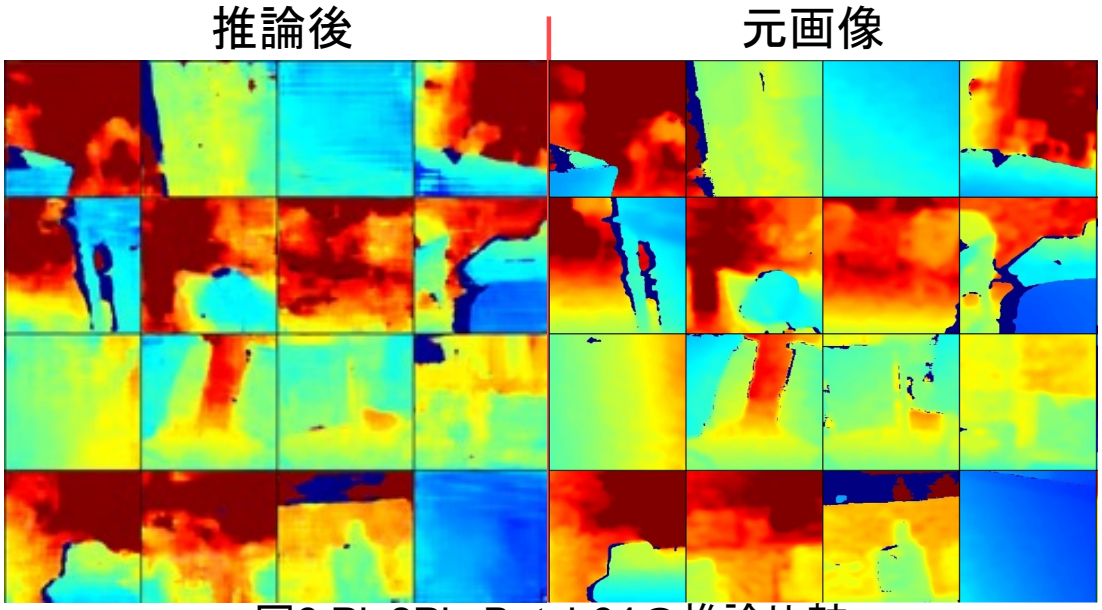

図6 Pix2Pix Batch64の推論比較

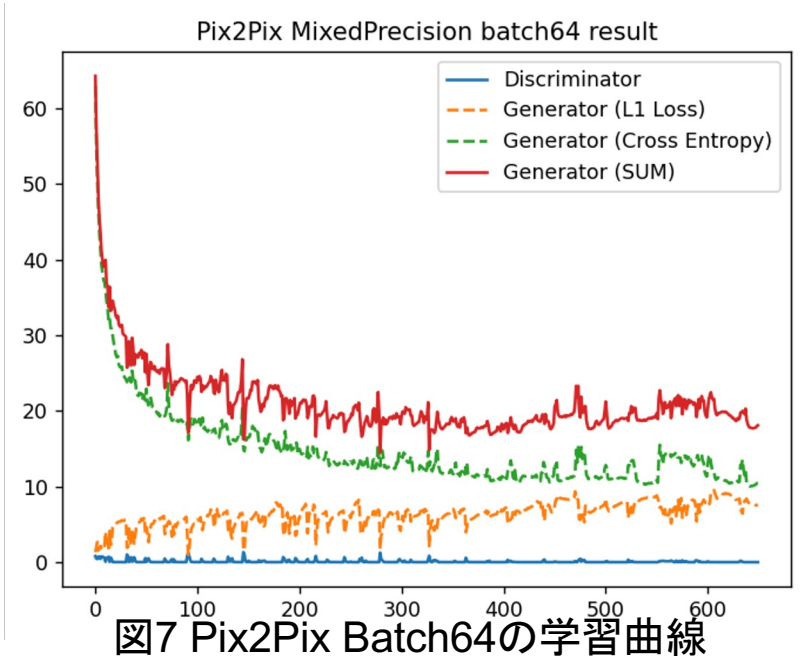

図7 Pix2Pix Batch64の学習曲線

学習データのリアルタイム推論による評価

カメラ入力に対しリアルタイム推論を行うことで，推論の有用性を検証した.

→６０FPSのカメラ入力に対しては処理落ちが発生することはなかった.

これを使用し，同一の動画を入力・比較することで，学習の比較を実行

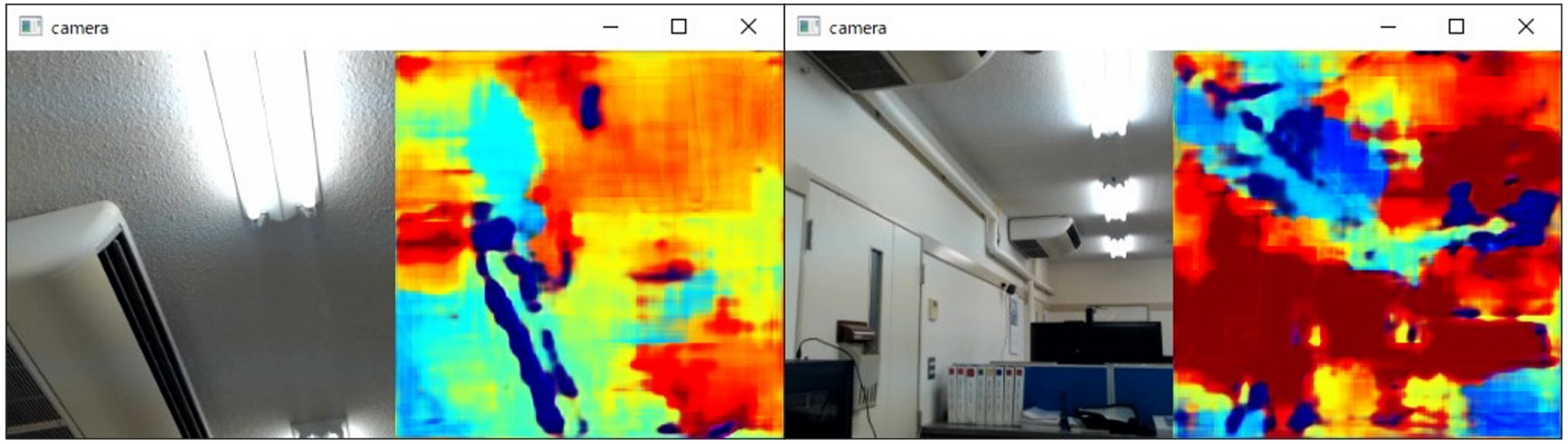

図8 リアルタイムカメラ推論画像

参考動画

この結果に対する考察

<u>良いと評価できる点</u>

・オブジェクト輪郭の目視

・色分布的上の着色正確性

<u>性能悪化要因</u>

・汎化性能不足

・電灯の周波数同期のチラつきによる画面明度の変化

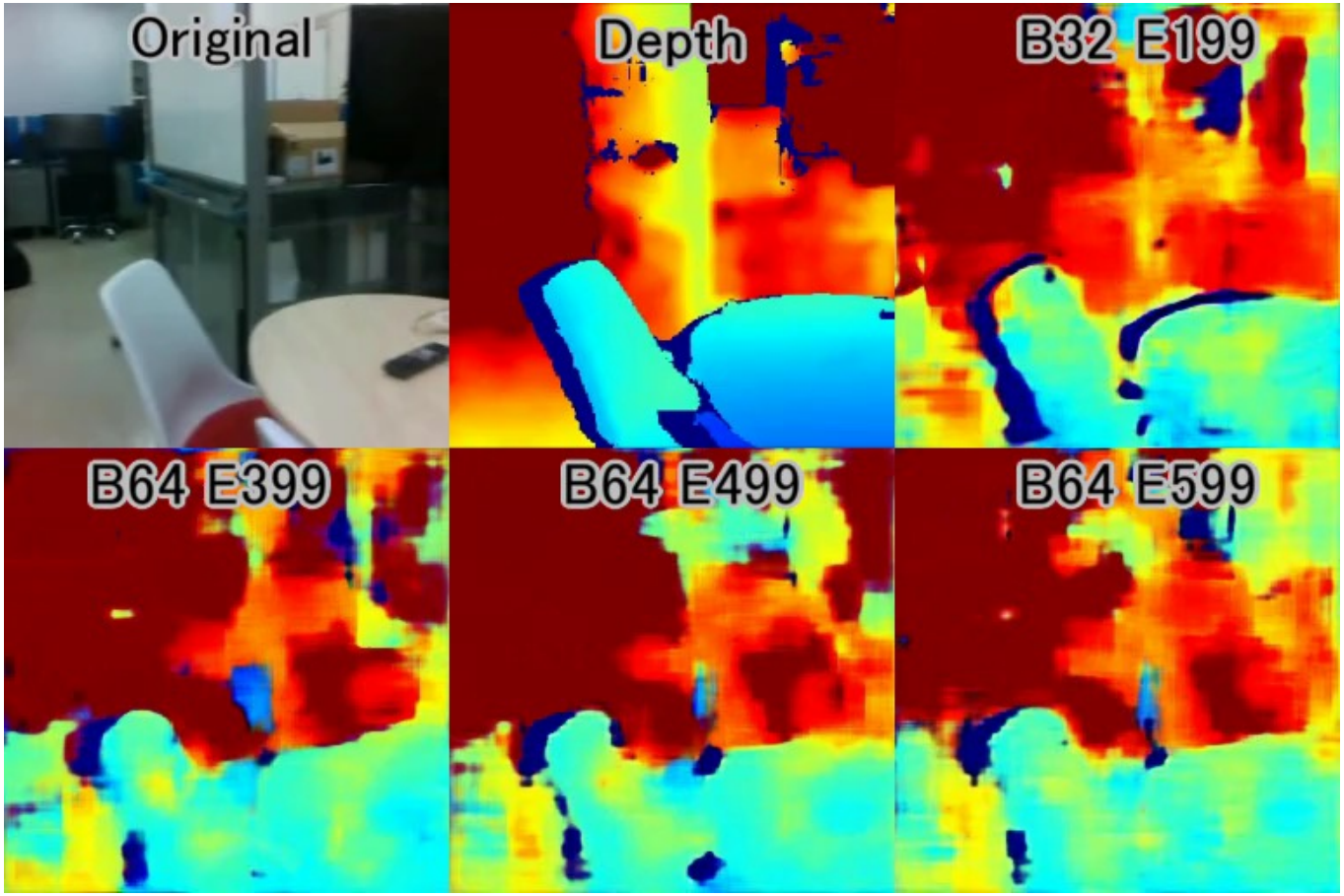

図9 推論動画

混合精度の推論精度に関して

検証映像での混合精度の推論結果が悪かったため，
テストデータを3ペア用意してMAE（L1 Loss)を算出し，比較を行った.

総じてBatch32での推論の方が誤差が少なく，精度がいいという結果となった.

→今回のケースでは混合精度手法での
精度向上には失敗してしまった.

表1 検証データのL1 Loss

エポック数	99	199	299	399	499	599
Batch32	33.99	32.48				
Batch64	38.92	27.96	32.10	33.14	33.10	32.80

エポック数	99	199	299	399	499	599
Batch32	43.25	42.43				
Batch64	47.04	51.54	48.70	50.85	47.88	50.73

エポック数	99	199	299	399	499	599
Batch32	34.21	30.40				
Batch64	43.33	39.79	35.03	44.23	39.50	39.05

まとめ

今回の試みである，RGB画像からRGB-D画像を生成することには成功した.
　しかし、精度や汎化性能ではまだまだ課題があると考える.

精度に関して
　今回，混合精度を用いた精度向上には失敗してしまった.

これは半精度による精度悪化と，Batchサイズを拡張したことによって学習に悪影響を与えてしまったと考えられる.

今後の展望
　さらなるデータセットの蓄積や，他の精度向上手法の適用によってどれほどの精度獲得ができるか注目したい.