広島県立総合技術研究所東部工業技術センター研究報告　No.37（2024）
技術ノート

深層学習による異常検知手法の簡単な比較（第５報）

敵対的生成ネットワーク（第１報）

廣川 勝久　佐々木 秀和　押野 純大　山形 亮太

Anomaly Detections using Deep Learning (Ⅴ)

Generative Adversarial Networks (Ⅰ)

HIROKAWA Katsuhisa, SASAKI Hidekazu, OSHINO Atsuhiro, YAMAGATA Ryota

我々はこれまで異常検知の手法として、正常信号のみを学習したオートエンコーダやボルツマンマシンが異常信号を検出できることを報告した。本報告では、敵対的生成ネットワークが、オートエンコーダなどの生成能力を向上させ、ボケの少ない異常検出モデルの学習に有効であることを示す。また、異常領域の検出精度を向上させる手法についても述べる。

キーワード：敵対的ネットワーク，オートエンコーダ，Generative Adversarial Networks, GANomaly ,U-Net

1. 緒　　言

センサーやカメラ、インターネットなどを利用して、製造装置、制御機器、気象、通信、交通などの異常を自動検知する試みが様々な分野で成されている。異常検知の開発では､通常得られるデータは正常なデータのみで、異常なデータが発生することは稀であり、発生が見込まれる特異な異常データを網羅的に開発時に取得することは不可能である。このため、正常データに対して一定の基準を設け､その基準から外れたものを異常としている。

一方、異常検知に深層学習を適用する場合、正常データと比較して異常データが少ないことから、教師あり学習では、正常データの過学習が発生することが予想される。また、学習した異常データ以外の異常データに対する判別が不確実となることも想定される。従って、異常検知への深層学習の適用には、正常データの学習のみの学習モデルにより、異常データを検出することが必要である。

前報では、生成ネットワークであるオートエンコーダ（**A**uto **E**ncoder: **AE**）[1, 2]やボルツマンマシン（**B**oltzmann **M**achine:**BM**）[3]を用いて正常信号のみの学習から異常信号が検出できることを報告した。これらのモデルでは、学習により正常信号を低次元の特徴値による分類するクラスタリングが行われ、学習後、異常信号の入力に対しては、入力の特徴値に近いクラスタに学習された特徴値から典型的な正常信号を生成する。生成された正常信号と入力した異常信号を比較することで異常部分の差分検出が行われる。しかし、これらのモデルでは、低次元化により汎化能力を向上させた結果、入力が正常信号であっても生成される信号にはボケが生じる欠点があり、異常信号領域のみを正確に検出できなかった。

本報告では､敵対的生成ネットワーク（**G**enerative **A**dversarial **N**etworks：**GAN**）[4]の実装により、オートエンコーダ単独学習の場合に比べ、ボケの少ない異常検出モデルが学習可能であることを示す。また、異常領域の検出精度を向上させる手法として、擬似的な異常を正常画像に加えた学習を行うことにより、正常画像の検出能力を低下させることなく、異常領域の検出能力が向上することについても述べる。

2.敵対的生成ネットワーク

2.1 敵対的生成ネットワークと異常検知

敵対的生成ネットワークは拡散モデルなどと同様に、学習データから、新しいデータを生成する深層生成モデルとして開発された。敵対的生成ネットワークでは、発生器と識別器の２つのネットワークが競合するように学習が行われる。学習後、敵対的生成ネットワークの発生器にランダムなデータを与えると、それに対応したデータが生成される。異常検知ではこの発生器部分をランダムな入力ではなく、異常信号の入力から正常信号を生成させようとするものである。

2.2 異常検知用敵対的生成ネットワーク

異常検知用敵対的ネットーワークのモデルにはGANomaly を用いた（**図 1**）[5]。このモデルの発生器には2次元畳み込みオートエンコーダが実装されている｡また識別器には､発生器のエンコーダ部分が使用されている。一般的な敵対的ネットーワークでは、発生器からの出力と学習データを識別器に入力し、発生器・識別器の学習を交互に行う。しかし、GANomaly においては、この学習方法では、学習後、入力したデータとは異なる学習データが発生器から出力されるGANomaly では発生器の入出力データを一致させるため、学習時の学習誤差が以下のとおり設定されている。

2024.9.27 加工技術研究部　受理

1. 発生器の入力データと出力データの差を学習誤差として、入出データを一致させる。

$$loss1 = |x - \hat{x}| \quad (1)$$

2. 発生器の出力$\hat{x}$を発生器のエンコーダに再度入力し、それぞれの潜在変数z、$\hat{z}$の差を学習誤差として計算する。

$$loss2 = (z - \hat{z})^2 \quad (2)$$

3. 入力データと出力データに対する識別器の潜在変数の差も学習誤差とする。

$$loss3 = [f(x) - f(\hat{x})]^2 \quad (3)$$

ここで $f(\cdot)$は識別器通過による非線形応答関数とする。最終的に GANomaly の学習誤差は

$$loss = w1 \times loss1 + w2 \times loss2 + w3 \times loss3 \quad (4)$$

により与えられる。ここで、$w1$、 $w2$、 $w3$ は各誤差に対する重み付けである。

加えて、GANomaly にスキップ接続を追加した Skip-GANomaly[6] についても比較検討を行った。Skip-

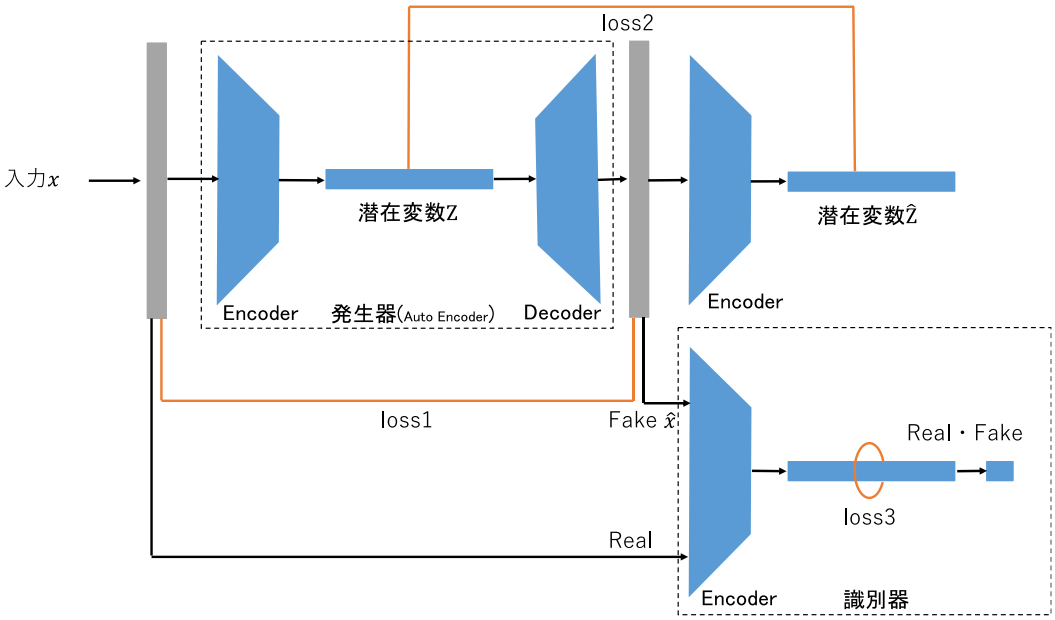


図 1　GANomaly の構造

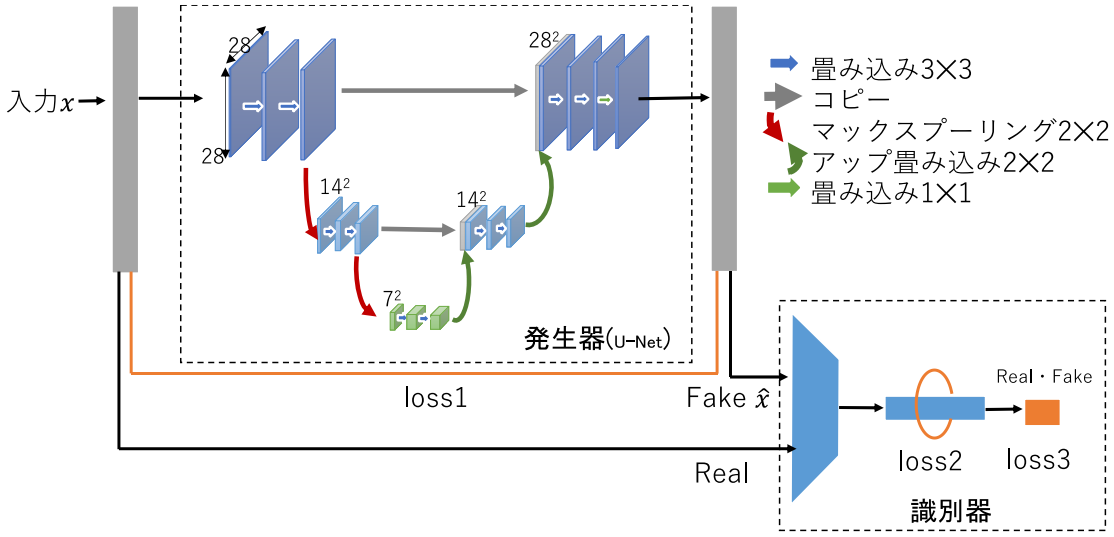


図 2　Skip-GANomaly の構造

GANomaly の発生器はオートエンコーダにスキップ接続の追加のみだが、本稿ではより高性能な U-Net の構造を用いた(図 2)。学習時の学習誤差には GANomaly のような発生器の潜在変数は用いない。

ネットワーク重みの更新には、まず発生器の入出力の差を誤差とする。

$$loss1 = |x - \hat{x}| \tag{5}$$

次に、識別器の潜在変数の差を学習誤差とする。

$$loss2 = [f(x) - f(\hat{x})]^2 \tag{6}$$

最後に、一般的な敵対的ネットーワークと同様に識別器の出力を学習データに近づける。

$$loss3 = \log[D(\hat{x})] \tag{7}$$

これら 3 つの誤差を足し合わせ、発生器の誤差として学習を行う。

$$loss = w1 \times loss1 + w2 \times loss2 + w3 \times loss3 \tag{8}$$

GANomaly、Skip-GANomaly の識別器に対する学習誤差は

$$loss = \log[D(x)] + \log[1 - D(\hat{x})] \tag{9}$$

により与えられる。

3. 異常検知用敵対的ネットワークの実装

3.1 学習用データ

敵対的生成ネットワークの評価のために、MNIST : Modified National Institute of Standards and Technology[7]のデータベースによる 6 万字のグレスケール手書き数字を学習させた。28×28 画素からなる MNIST の 2 次元手書き数字を敵対的生成ネットワークの発生器と識別器に 100 枚単位で入力し学習誤差を修正した。学習データには、事前に平均 0、分散 1 の画像に変換したものを用いた。

3.2 GANomaly の実装

GANomaly では発生器からの出力を再度エンコーダに入力する必要があるため、同じエンコーダを再度使用している。

```
class Generator(torch.nn.Module):

    def __init__(self):
        super().__init__()
        self.encoder = Encoder()
        self.decoder = Decoder()
    def forward(self, x):
        z = self.encoder(x)
        x = self.decoder(z)
        z2 = self.encoder(x)
        return x, z, z2
```

2 次元畳み込みオートエンコーダについては、以前報告したものを用いた[2]。ただし潜在変数のサイズは 1024 とした。識別器のエンコーダは発生器のエンコーダ同じであり、潜在変数と識別値の 2 つの値を出力する。

```
class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = Encoder()
        self.conv = nn.Conv2d(zsize, 1, 1, 1, 0,
                              bias=False)
    def forward(self, x):
        z = self.encoder(x)
        x = self.conv(z)
        x = torch.sigmoid(x)
        return x.view(-1), z
```

発生器の学習では、発生器と識別器の両方を使用する。学習データを発生器に入力し、その出力を識別器に入力する。合わせて識別器には学習データの入力からも出力を得る。発生器と識別器の入出力から(4)式の誤差を求めネットワーク重みの更新を行う。

```
        netG.zero_grad()
        fake, z, zdash = netG(real_imgs)
        _, zf = netD(fake)
        _, zr = netD(real_imgs)
        errG = (
            w1 * criterion2(zf, zr)
            + w2 * criterion3(fake, real_imgs)
            + w3 * criterion2(z, zdash)
        )
        errG.backward()
        optimizerG.step()
```

この時、誤差関数には以下を用いた。

```
criterion2 = nn.MSELoss(reduction="mean")
criterion3 = nn.L1Loss()
```

識別器の学習は(9)式により、学習データと発生器からのデータの分類ができるように学習が行われる。すなわち、学習データが入力された場合は 1 を出力し、発生器からのデータについては 0 の出力に近づくよう、ネットワークの重み修正が行われる。

```
        netD.zero_grad()
        output, _ = netD(fake.detach())
        errD_fake = criterion(output, zero_labels)

        output, _ = netD(real_imgs)
        errD_real = criterion(output, one_labels)

        errD = errD_real + errD_fake
        errD.backward()
        optimizerD.step()
```

3.3 Skip-GANomaly の実装

Skip-GANomaly の識別器は GANomaly と全く同じものを実装した。一方、発生器については、拡散モデルで報告した U-Net[8]を微修正後、実装した。

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

```
        self.layer1 = DownConv(classter1,
                               classter2)

        self.layer2 = DownConv(classter2,
                               classter3)

        self.layer3 = UpConv(classter3, classter4,
                               classter5, flag=0)

        self.layer4 = UpConv(classter3+classter5,
                             classter5, classter6, flag=0)

        self.layer5 = UpConv( classter2+classter6, classter6,
                            channels, flag=1)

    def forward(self, img):
        org1, down1 = self.layer1(img)
        org2, down2 = self.layer2(down1)
        up3 = self.layer3(down2, down2, 0)
        up4 = self.layer4(up3, org2, Layer3size)
        img = self.layer5(up4, org1, Layer4size)
        return img
```

(8)式に従い発生器の学習を行う。

```
        netG.zero_grad()
        fake        = netG(real_imgs)
        _     , zr = netD(real_imgs)
        outf, zf = netD(fake)

        errG = ( w1 * criterion2(zf, zr)
                 + w2 * criterion3(fake, real_imgs)
                 + w3 * criterion (outf, one_labels)
        )
        errG.backward()
        optimizerG.step()
```

発生器の学習には 3 種類の誤差関数を用いた。

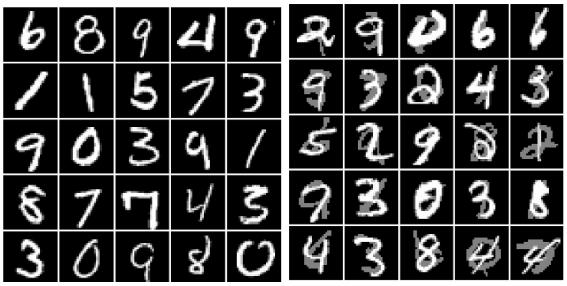
```
criterion = nn.BCELoss()
criterion2 = nn.MSELoss(reduction="mean")
criterion3 = nn.L1Loss()
```

4. 異常検知能力評価結果

4.1 GANomaly の異常検知能力

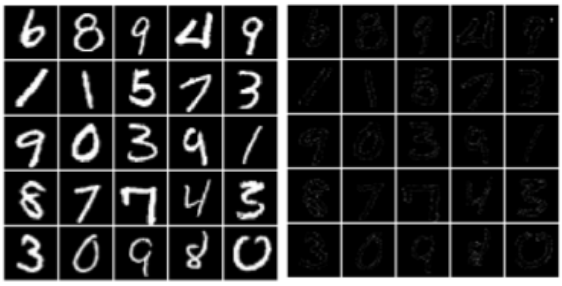
敵対的生成ネットワークの異常検知能力を評価するために、評価用データとして**図 3** に示す 2 種類の画像を用いた。正常データとして MNIST の画像を、異常データとして正常画像に、50%程度に輝度を低下させた MINST 画像を重ねた異常画像を用意した。GANomaly に正常画像 6 万枚を 50 回学習させた後、発生器に正常画像と異常画像を入力し、発生器が生成する画像により評価を行った。**図 4 (a)** は異常画像から発生器が生成した画像を示す。また**図 4 (b)** は入力画像と生成画像との差分による画像である。実験結果から、GANomaly による生成器の学習では、正常画像に全くボケを生じない学習が可能であった。次に、同発生器に異常画像を入力した結果を**図 5** に示す。**図 5 (a)** の生成画像を入力画像と比較すると完全では無いが異常部分の修正が行われている。その結果**図 5 (b)** の差分画像には修正された領域が表示され、異常領域の検知が行われた。

4.2 敵対的生成ネットワークの異常検知精度向上



(a) (b)

図 3 入力画像
(a) 正常画像, (b) 異常画像



(a) (b)

図 4 検知結果 (正常画像)
(a) 生成画像, (b) 差分画像

(a) (b)

図 5 検知結果 (異常画像)
(a) 生成画像, (b) 差分画像

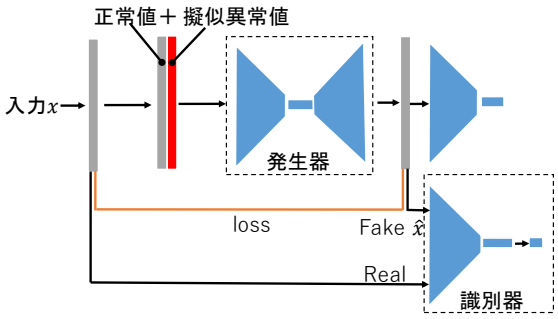
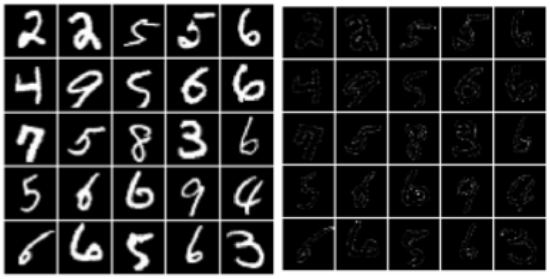


図 6 異常画像の学習

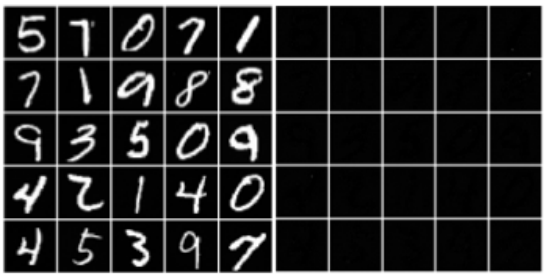
GANomaly の正常画像のみの学習によって、異常が検知できることを示したが、**図 5 (a)** の生成画像では、異常領域すべての修正が行われてはいない。このため異常領域の検出精度を向上させるため、**図 6** に示すように正常な



(a)　　(b)

(c)　　(d)

図 7　検知結果(GANomaly)
(a)正常画像からの生成画像, (b)正常画像との差分画像
(c)異常画像からの生成画像, (d)異常画像との差分画像



(a)　　(b)

(c)　　(d)

図 8　検知結果(Skip-GANomaly)
(a)正常画像からの生成画像, (b)正常画像との差分画像
(c)異常画像からの生成画像, (d)異常画像との差分画像

入力画像に擬似的に異常値を付加し、発生器に入力する学習を行う。この時、発生器の出力と正常画像との差を誤差として発生器の学習を行う。

この手法により GANomaly の学習結果について評価した。**図 7** は学習後の GANomaly に正常画像と異常画像を入力し、生成させた画像を示す。正常画像からの生成画像には、一見してノイズなどは無いが、差分画像**図 4**(b)と**図 7**(b)を比較すると正常画像の入力に対する生成にはわずかにボケが生じている。一方異常画像の入力に対する生成画像には異常領域が全く残っていない。このため、異常領域が正確に検知されるようになり、**図 7**(d)からは異常として付加された画像の手書き文字が読み取れる。

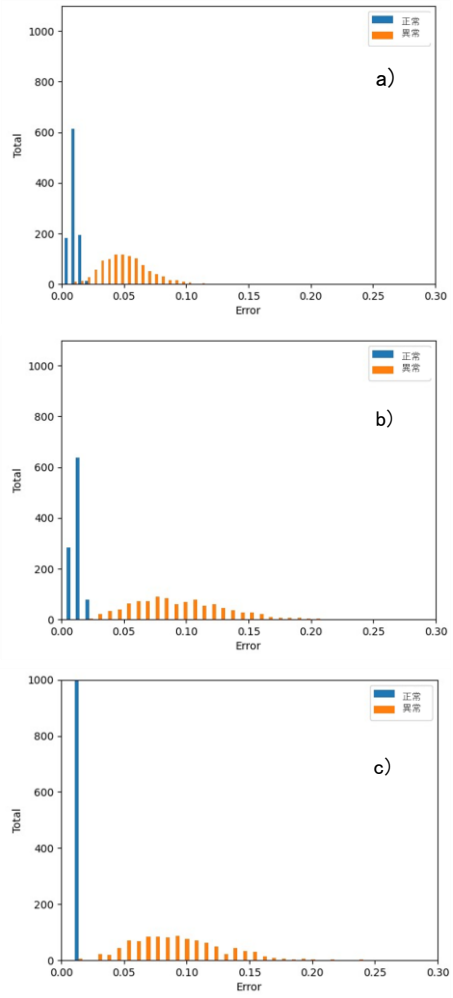


図 9　異常値の分布
(a)正常画像を学習した GANomaly
(b) 異常画像を学習した GANomaly
(c) 異常画像を学習した Skip-GANomaly

正常な入力画像に擬似的に異常値を付加した Skip-GANomaly の結果を**図 8** に示す。正常画像を入力した GANomaly の結果（**図 7**(b)、**図 8**(b)）と比較すると Skip-GANomaly では正常画像の入力に対してボケが抑制されている。また異常画像の入力についても Skip-GANomaly は GANomaly と同様に生成画像には異常領域が全く残っておらず、異常領域の検知も正確に行われている。

最後に、入力画像の全画素値の合計に対する差分画像の全画素値の割合をグラフ化することにより比較する。横軸は入力画像に対する差分画像の割合を示し、縦軸はその枚数を示す。ここでは、正常画像、異常画像、それ

ぞれ 1000 枚を学習済みの発生器に入力し、比較した。図 9(a)は正常画像を学習したGANormalyの場合のグラフを示す。正常画像に対応する差分画像の画素値の合計は値が小さく、狭い領域に分布している。一方異常画像については、画素値の合計は広く分布し、大半は正常画像より大きな値となった。異常画像を学習した GANormaly の結果を図 9(b)に示す。図 9(a)と比較して異常画像に対する差分画像の値が大きく、異常領域の検知能力が向上した結果となった。Skip-GANomaly の場合(図 9(c))、異常画像の分布は GANormaly のそれと差異は無いが、正常画像に対する分布は GANormaly より狭いため、図 8(b)のボケが抑制を裏付ける結果を得た。

5. 結　　言

敵対的生成ネットワーク(**G**enerative **A**dversarial **N**etworks:**GAN**)であるGANomalyとSkip-GANomaly実装し、ボケの少ない異常検出モデルが学習可能であることを示した。実験では、いずれの学習モデルにおいても、正常画像に対応する差分画像の割合の最大値以上のしきい値を設定することにより、大半の異常画像を検知することが可能であると考えられる。さらに異常領域までも正確に検知するには擬似異常値を使った学習が効果的であった。ただし、正常画像のみを学習した Skip-GANomaly については U-Net の学習能力が高く、異常を検知できなかった。

文　　献

1) 廣川 勝久：広島県立総合技術研究所東部工業技術センター研究報告，深層学習による異常検知手法の簡単な比較（第 1 報）35（2022）.
2) 廣川 勝久：広島県立総合技術研究所東部工業技術センター研究報告，深層学習による異常検知手法の簡単な比較（第 2 報）36（2023）.
3) 廣川 勝久：広島県立総合技術研究所東部工業技術センター研究報告，深層学習による異常検知手法の簡単な比較（第 4 報）36（2023）.
4) 廣川 勝久：広島県立総合技術研究所東部工業技術センター研究報告，強化学習における各種手法の比較（第 3 報）35（2022）.
5) Samet Akcay1, Amir Atapour-Abarghouei1, and Toby P. Breckon : GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. arXiv:**1805.06725** (2018).
6) Samet Akçay, Amir Atapour-Abarghouei, and Toby P. Breckon : Skip-GANomaly: Skip Connected and Adversarially Trained Encoder-Decoder Anomaly Detection. arXiv:**1901.08954** (2019).
7) THE MNIST DATABASE of handwritten digits : http://yann.lecun.com/exdb/mnist/
8) 廣川 勝久： 広島県立総合技術研究所東部工業技術センター研究報告，ノイズ除去拡散確率モデル（第１報） 深層学習による画像の領域分割（第 1 報）37（2024）.