第10章　おっぱいのモザイク解除にチャレンジ

※注：本書は技術書典6で頒布した『DeepCreamPyで学ぶモザイク除去～アソコもAIもディープにラーニング～』のおまけというか余興の内容である。この章の内容は技術書典の頒布物には含んでいない。

この本では長い時間をかけて「モザイクマスター」への道を歩んできた。今、まさにその地に立ったのである。まずはそのことにおめでとうと言いたい。非常にハードな内容だったがよく頑張ったね。

この章では余興として、楽しくかつチャレンジングな例を紹介したい。**おっぱいのモザイク解除**である。楽しむためなので硬い解説はかなり省略する。硬いのは下半身だけで十分だ（大事なことだが、太ももの筋肉の話だ）。

こしあん王国

（※茶番注意）

あるところにこしあん王国があった。そこには絶対的な権力を持つアン女王がいる。アン女王は気まぐれで時々突拍子もない政策を出すことがある。あなたは宮廷専属のエンジニアとなり、機械学習の力を借りて王国の困難に立ち向かわなければならない。

アン女王「おい、おぬし、どの女を見ているのだ？」

あなたがテレビを見ていたときだった。テレビにはナイスバディな美少女アイドルがいた。

> あなた「いえ、ただテレビを見ていただけであります。国政の把握は大事であります」
> アン女王「ほう、おぬしの目線はその女の胸元に釘付けなようだが……」
> あなた「いえ、そのようなことは決してございません」
> アン女王「その女のことは聞いておるぞ。最近流行のアイドルとやらではないか」
> あなた「はい、かわいいですよね。さすが我が国のトップアイドルだけあります。これなら我が国も他国に引けを取りません」

アン女王は激怒した。必ず、かの邪智暴虐のおっぱいを除かなければならぬと決意した。アン女王には巨乳がわからぬ。アン女王は、貧乳の王である。「女王陛下は世界一高貴ですよ」とおだてられながら、召使いに囲まれて暮らして来た。けれども胸の大きさに対しては、人一倍に敏感であった。

> アン女王「おぬしは、わらわの高貴な胸より……その…下賤な平民のバインバインの乳のほうが好きと申すか？　そうなのか？？」

迂闊だった。最近忙しくてアン女王のお世話をしていなかったのだ。アン女王はご機嫌が斜めだと貧乳コンプレックスが炸裂するのだ。

> あなた「女王陛下のお胸が世界一であります！　Yes, your Majesty！」
> アン女王「そうは言っても目が泳いでおるぞ。男とはそういう生き物なのじゃな……待てよ」

嫌な予感がする。女王の突拍子もない思いつきだ。

> アン女王「このバインバインの乳がなければ、おぬしはわらわともっと遊んでくれるのじゃな。そうじゃ。テレビに規制を入れてるのじゃ。下賤のものの乳は全てモザイクをかければよいのじゃ。世のすべての人はわらわの高貴な胸だけ見ていればよいのじゃ。これはお触れとして即実行じゃな。」

悪い予感が的中してしまった。しかし、これはまずい。アン女王は貧乳でも、こしあん王国の国民はおっぱいが大好きなのである。これは国民の士気に関わる。

> あなた「私はそれでよくても、民の士気に関わります。王国の生産性を落としかねません」
> アン女王「なんじゃと、それは本当か！　それなら、余計規制を実行せねばならぬな。このような不健全な乳がなくても素晴らしい生産を上げられるよう民を教育するのじゃ。女王の聖なる威光を示すのじゃ」

埒が明かない。ここは全面降伏で女王をなだめる作戦でいこう。

> あなた「大変申し訳ありませんでした。私の不健全で下賤な心が、かのようなバインバインのおっぱいにうつつを抜かしてしまいました。つきましては女王陛下の寛大なる……」

それもつかの間、既にその場にアン女王の姿はなかった。規制の話が歩み始めたのである。女王のモットーは即断即決だ。この方針は、話が難しい政治上の問題に対して非常に効果を発揮するのだが、女王の気まぐれに対してはデメリットでしかない。どうしよう。

そこに話を聞いていたメイド長が通りかかった。

> メイド長「あらあら、大変でしたね」
> あなた「ええ、女王陛下の気まぐれには困ったものです」
> メイド長「そうは言っても、陛下があそこまで心を開いてくれるのはメイドでもほとんどいませんよ。昔はメイドに対してももっと他人行儀でどこか遠くを……」
> あなた「はぁ……国民からの反発どうしよう」
> メイド長「ああ、その話ですか。私が話を通しておきましたので、規制を解除する装置を作ってください。表向きには規制するということにしていますが、抜け道を作っておきますので。」
> あなた「え？　なんですかその話は。規制するという話じゃなかったのですか」

なにやらややこしいことになってきた。

> メイド長「我が国の放送における健全な表現のための規制――わかりやすいように、おっぱい規制と呼びますね、は明日付で女王陛下のお触れとして公布されます。これはしばらくの準備期間をおいたあと、半年間の試験運用として施行されます」
> あなた「つまり、半年待てば元に戻るというわけですね」
> メイド長「もちろんそれもできます。しかし、もっと積極的なことをやっていただきたいのです。」
> あなた「というと？」
> メイド長「おっぱい規制と同時に、規制を解除する装置を民間と共同で秘密裏に開発します。そしてこれを民間から販売します。これにぜひ参加していただきたいのです。」
> あなた「え、そんなのいいんですか？　そんなことしたら処刑対象になってしまうのでは」
> メイド長「おっぱい規制は女王陛下直属の部署で行います。そして、これは秘密ですが、解除装置も同じ部署で開発します。つまり、規制と開発の管轄が同じなので、解除装置の販売を罰することはできません」

なんじゃそりゃ、そんなことあっていいのかという話だ。

> あなた「でもそうまでして実施する意味ってなんですか？」
> メイド長「一つは陛下に貧乳コンプレックスを克服していただくためです。あのコンプレックスは王国の弱点になりかねません」

確かにそれはそうだ。でも規制したら余計悪化するような…。

> メイド長「二つ目は、これは規制の解除の話ですが、先程おっしゃられたとおりに国民感情です。いくら女王陛下が貧乳だからといっても、人間は大きなおっぱいが好きなものです。」

話のわかるメイド長でよかった。We Love Big おっぱいだ。

> メイド長「そして三つ目は、これが半ば本題なのですが……」

おっ、なにやら深い話になってきた。たかがおっぱいでどれだけ話をするのだろう。

> メイド長「王国の情報技術戦略、そして軍事戦略と関係します。モザイクのような規制の解除技術は、情報戦略や軍事面で有望視されています。以前から開発の話はありました。」

これは言われてみれば確かに、な視点だ。今までそういった視点で見たことがなかった。

> あなた「あー確かに。おっぱい程度なら誰も気づかないでしょうが、偵察や情報戦で有効ですよね。特に見えない場所からより鮮明を抜き出す技術があれば、他国より優位に立てます。」
> メイド長「流石ですね。そのとおりです。そしておっぱいだから誰も気づかないという部分も含めて正解です。」
> あなた「ほう、それは斬新ですね」
> メイド長「こういった技術は表向きに開発すると、軍拡を警戒されて他国からマークされたり反発を招い

> たりします。そこで、おっぱい規制というわけです。テレビに出てくるおっぱいを規制するなんて話なら、他国はバカにして笑いものにするだけです。」
> あなた「女王陛下には申し訳ないですが、ずいぶん深いことを考えてらしたのですね」
> メイド長「ええ、そして解除装置の販売には『王国の新技術の開発協力』という形で、補助金を出すようにします。そうすれば国民負担は一切ありません」

女王の戯言を国策に転用するとはこのメイド長、末恐ろしい存在だ。

> あなた「つまり、規制というのは建前で、実質投資というわけですか？」
> メイド長「そういうことになります」
> あなた「それで私にモザイク解除装置を開発してほしいと」
> メイド長「そうです。おそらく適任かなと。私も詳しいことはわからないのですが、きかい…がくしゅう？でしたっけ、それを使うとモザイクが解除できると聞いたことがあります」

確かにそのとおりだ。DeepCreamPyがモザイク解除に使えるというのはインターネットで一時期話題になった。そして背景となっている技術はP-Convであると本で読んだことがある。幸いその実装は知っているので、頑張れば開発できる。

> あなた「なるほど、それなら私の専門というわけですね。似たような技術は聞いたことあるので、完全な除去は無理かもしれませんが、できる限りのことはやってみます」
> メイド長「快諾していただいて、どうもありがとうございます。アン様も喜ばれると思いますよ」

なんかメイド長に乗せられてしまった感がすごいが、おっぱいでそこまで語られたからにはやらなきゃ男の恥だ。

> あなた「最後に一ついいですか？」
> メイド長「はい、なんでしょう？」
> あなた「機械学習でモザイク解除ができるとどこで知ったのですか？　あまり普通の人は知らないと思いますが」
> メイド長「夜の情報収集も含めて、メイドのお仕事ですから♡」

このメイド長を敵に回すのだけはやめよう、と思った。

> メイド長「では、明日から頑張ってくださいね。王国の国民感情はあなたの手に全てかかっていますよ！」

すっかり忘れていた。ヘマをすると国民の怒りが全て自分の身に降り注ぐのだった。

――きかいがくしゅうの　ちからを　かりて　おうこくの　こんなんに　たちむかおう。

※茶番終わり

実は難しいおっぱいのモザイク解除

さて、おっぱいのモザイク解除をするわけになったのだが、この本は健全なのではだけたおっぱいのモザイク解除ではない（重要）。この章で扱うのは、**全年齢の健全なイラストの胸部に服の上からモザイクをかけて、それを解除するというタスク**である。この**服の上からという要素が、逆にモザイク解除を著しく難しくしている**。

きちんと試していないが、実ははだけたおっぱいのモザイクを解除するほうが、服の上からのモザイク解除より格段に楽だと思われる。服の上からのモザイク解除を試してみたのだが、U-Netを使っても鮮明な出力は厳しかった。もし鮮明な解除を期待していたのなら大変申し訳無い。なぜだろうか。

脱がせたおっぱいというのはキャラ問わず似たような肌色である。同じような色だから、少ないデータ数でも比較的学習が進みやすい。しかし、服の上からだと違う。**服はキャラによって違うし、胸の大きさも違うし、装飾品やリボン**（これが厳しかった）もついている。これらを学習するには大量のデータが必要だ。今回は、**胸部の領域のアノテーションがあるイラストの画像データセットがなかったので、一から自分でデータセットを作った**（後述）。全てを自分でやるには4000枚程度が限度だった。もし**脱がせていたらちゃんと学習できたのかもしれない**が、全年齢対象にする以上脱がせられないので、データ量と二重の意味でこれが限界だった。後ほど考えられる対策を書いていくので、ぜひ著者の無念を晴らしてほしい。

ポイント：おっぱいのモザイク解除は服の上より脱がしたほうが多分楽だが、大人の事情でできない。

おっぱいデータセット

この章を書くにあたり、おっぱい画像の領域を集めたデータセットというのを私が作った。次のような画像のデータセットである。

- 全年齢対象かつ女性キャラが描かれているイラストが対象
- おっぱいの領域に対して**境界箱**（Bouding Box）のデータ付けをした。これは全て私が行った。

境界箱とは、物体検出で用いられる物体の位置を表す箱である。このような感じである。なお、このイラストは[*1]をソースとしている。

境界箱とはこのようなエリアの指定である。おっぱいの領域をひたすら自分がタグ付け（専門用語ではアノテーション作業という）した。端から見れば相当シュールだったろうが、いろんなおっぱいが見れてとても楽しかった。世界一楽しいアノテーション作業だろう。

データ元はPixivからである。全年齢対象かつ「おっぱい」タグで検索してダウンロードし、アノテーション作業をした結果（長くつなげた漫画のような縦横比が極端な画像などは除外している）、**4264件**の画像が得られた。ただし、イラストが非公開になったり、ダウンロードできなくなると細かい枚数は変わる可能性はある。

また、元イラストの著作権に配慮し、データセットとして公開して入るものの、元画像のPixivURLを記載し、利用者がダウンロードする形で公開している（ImageNetと同じ方式）。当然、元のイラストの著作権はイラストレーターが保有しているため、**jpegファイルを一式をzipファイルに固めて、不特定多数に公開するなど著作権に抵触する行為は、絶対に行わないように**すること。以下のようなJSON方式で記載している。


```
    {
        "name": "72809999_p0_master1200.jpg",
        "filepath": "oppai_dataset/images/72809999_p0_master1200.jpg",
        "id": 72809999,
        "size": {
            "width": 600,
            "height": 848
        },
        "image_url": "https://i.pximg.net/c/600x1200_90/img-
master/img/2019/01/23/23/28/26/72809999_p0_master1200.jpg",
        "original_url": "https://www.pixiv.net/member_illust.php?
mode=medium&illust_id=72809999",
        "bounding_boxes": [
            {
                "height": 212.79998779296875,
```

```
                "width": 286.4000244140625,
                "left": 57.4000244140625,
                "top": 258.20001220703125
            }
        ],
        "n_bounding_boxes": 1
    },
```


URLは公然と周知された情報だし、境界箱の数値データは全て私がつけたものなので、この方式での公開なら何ら問題がない。アノテーション作業にはVoTTという専用のソフトを使った（手動でのアノテーション作業には変わりないが、マウスクリックで座標を勝手に記録してくれるので使い勝手が良い）。これなら1枚数秒程度でできき、1人で4000枚オーバーというのも現実的な作業時間になる。

以下のURLから利用できる。

https://github.com/koshian2/DeepCreamPyBook/tree/master/Oppai/oppai_dataset

ポイント：おっぱいデータセットというのを作った

おっぱいデータセットのダウンロード

ではおっぱいデータセットの画像をダウンロードしてみよう。ColabでダウンロードしてGoogle Driveに保存する例を示す。ただし、Pixivから画像をダウンロードする際に、**パスワードを平文で保存することになるので取扱に注意しよう（特にイラストを投稿しているような大事なアカウントの場合）**。

Pixiv IDはログインメールアドレスではない。「Pixivのユーザー画面→基本設定→pixiv ID」で表示される半角英数が対象となる。

Colabでのダウンロード

Colabを起動する。インスタンスはなんでも良い（CPUで良い）。まずはおっぱいデータセットのコピーのために「Subversion」をインストールする。

```
!apt install subversion
```

この本のリポジトリの「おっぱいデータセット」の部分だけコピーする。

```
!svn export https://github.com/koshian2/DeepCreamPyBook/trunk/Oppai/oppai_dataset --
force
```

pixivpyというPixiv APIのラッパーをインストールする。

```
!pip install pixivpy --upgrade
```

次のコードを実行する。成功するとダウンロードが開始される。ここで「your pixiv user id」と「your pixiv password」にはPixiv IDとパスワードを入力する。正しくログインされればダウンロードが始まる。

```
from oppai_dataset.download_images import *

set_login_attribute("your pixiv user id", "your pixiv password") # ここにID名、パスワ
ードを入力する
session = login()
download_all(session, "oppai_dataset/oppai_meta.json")
```

Google Driveにコピー

訓練するたびにいちいちダウンロードするのはPixivのサーバーに負担をかけてしまうので、一旦Zipファイルに保存してGoogle Driveに自分専用に保存しておく。

Google Driveのルート直下に「oppai」というフォルダを作っておこう。これはブラウザからできるので省略。いつものようにColabでマウントをする。

```
from google.colab import drive
drive.mount("gdrive")
```

ダウンロードした画像をZipファイルに固めよう。ファイルが多くログが大変なことになるので/dev/nullで捨てる。

```
!zip -r oppai_dataset.zip oppai_dataset > /dev/null
```

これでコピー完了だ。

```
!cp oppai_dataset.zip gdrive/My\ Drive/oppai/oppai_dataset.zip
```

Google DriveにコピーしたZipファイルをダウンロードして中を確認してみよう（Google Driveへの反映には若干ラグがある）。imagesフォルダに画像が大量に格納されているはずである。

Google DriveからColabで利用する場合

訓練などでインスタンスがリセットされたケースでデータセットを利用する場合は、Google Driveからコピーする。同じくGoogle Driveをマウントしよう。

```
from google.colab import drive
drive.mount("gdrive")
```

インスタンスのリセットを再現するには、Google Driveにzipファイルがコピーされたのを確認してから、「ランタイム→全てのランタイムをリセット」をクリックする。Google Driveをマウントしたらzipをコピーする。

```
!cp gdrive/My\ Drive/oppai/oppai_dataset.zip oppai_dataset.zip
```

Zipを解凍する。ログが流れまくるのでまた/dev/nullしよう。

```
!unzip oppai_dataset.zip > /dev/null
```

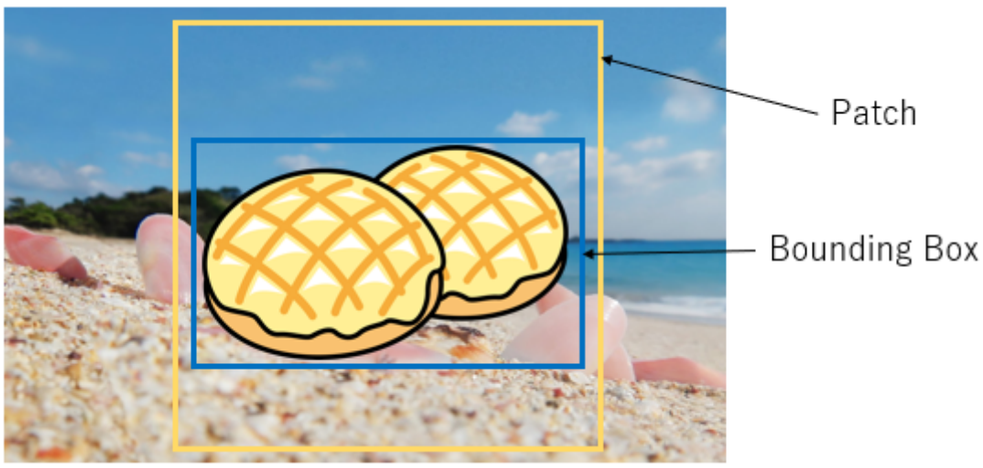
これで「oppai_datset/images」というフォルダにダウンロードした画像が全て再現される。Colabの左側のメニューをクリックし、ファイルからimagesフォルダを確認してみよう。ファイルをダブルクリックすると画像がポップアップする。

全体の方針

もう最後なので、実装の細かな詳細をくどくど説明するようなことはしたくはない。大まかな方針だけ説明する。

1. おっぱいのBounding Boxを基準に、拡大したパッチを作る

画像全体をP-Conv U-Netにかける方法もあるが、解像度が大きくなりがちなので、計算効率が悪い。本当に修正が必要な部分は限られているので、その周囲だけクロップして小さいパッチ単位で訓練したほうが速い（出力画質についてはそこまで変わるわけではないが、無駄な計算が少なくなる）。

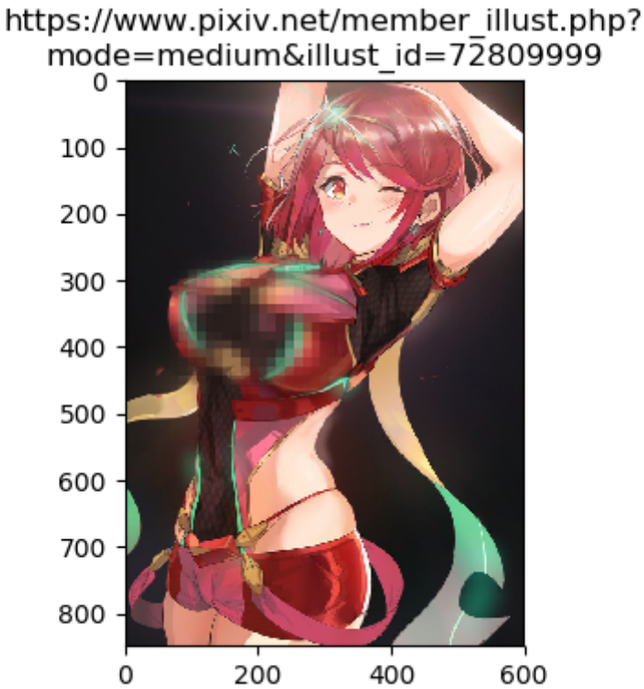


パッチとBounding Boxはこのような関係だ。なお、DeepCreamPyもこのようなパッチを作っており、パッチの作成にはDeepCreamPyのコードを元に改変している。ちなみにこれに気づいたのが、技術書典開催の2日前だったのでかなり青ざめた記憶がある（慌ててコードを書き直して訓練し直した）。

解像度にもよるが、240×320の縦長解像度で全体を一括で訓練した場合、TPU環境で1エポック8〜9分かかった。しかし、パッチを作って256×256の正方形の解像度で訓練した場合、パッチで画像のサイズが増えても（1枚の画像に複数のBounding Boxがあるケースが存在するため）、1エポック3〜4分に短縮することができた。これはなかなか良い高速化だ。

2.パッチ単位でモザイクを入れる

パッチ単位でモザイクを入れる。元画像：[*2]

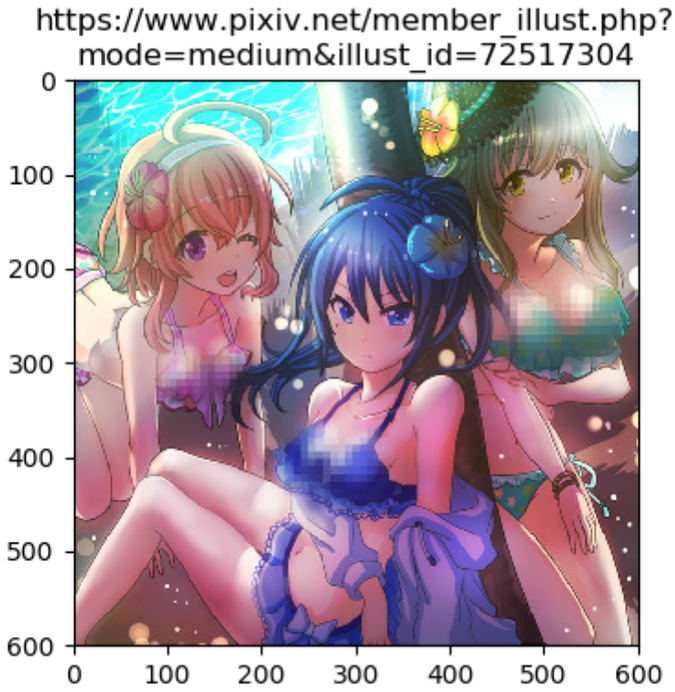


ガウシアンブラーをかけて、縮小→拡大でモザイクは再現できる。おっぱいの中央部だけモザイクをかけるように、マスク画像にもガウシアンブラーをかけてアルファブレンディングしている。

詳細な実装は以下のファイルにある。「add_mosaic」という関数がそれだ。

https://github.com/koshian2/DeepCreamPyBook/blob/master/Oppai/libs/utils.py

パッチが複数ある場合は次のようになる。元画像：[*3]



こしあん王国の国民はこんな世界を見ることになるのだ。確かにこれは国民の士気に関わる。

3.ネットワーク構成

よくあるU-Netだ。conv層はP-Convを使っている。「P-Conv U-Net」とでもいうべきだろうか。

```
def conv_bn_relu(image_in, mask_in, filters, kernel_size,
                 downsampling=1, upsampling=1, act="relu",
                 concat_img=None, concat_mask=None, reps=1):
    assert not (concat_img is None)^(concat_mask is None) # XORは常にFalse
    # Upsamplingする場合
    if upsampling > 1:
        conv = layers.Lambda(upsampling2d_tpu, arguments={"scale":upsampling})
(image_in)
        mask = layers.Lambda(upsampling2d_tpu, arguments={"scale":upsampling})
(mask_in)
    else:
        conv, mask = image_in, mask_in
    if concat_img is not None and concat_mask is not None:
        conv = layers.Concatenate()([conv, concat_img])
        mask = layers.Concatenate()([mask, concat_mask])

    for i in range(reps):
        stride = downsampling if i == 0 else 1
        # strideでダウンサンプリング
        conv, mask = PConv2D(filters=filters, kernel_size=kernel_size,
                             padding="same", strides=stride)([conv, mask])
        # Image側だけBN->ReLUを入れる
        conv = layers.BatchNormalization()(conv)
        if act == "relu":
            conv = layers.Activation("relu")(conv)
        if act == "tanh":
            conv = layers.Activation("tanh", name="unmasked")(conv)
    return conv, mask

def create_train_pconv_unet():
    input_image = layers.Input((256,256,3))
    input_mask = layers.Input((256,256,3))
    input_groundtruth = layers.Input((256,256,3))

    # マスクを0-1にして広げる
```

```
    expanded_mask = layers.Lambda(lambda x: 1.0-K.sign(1.0-x))(input_mask)

    ## U-Net
    # Encoder
    conv1, mask1 = conv_bn_relu(input_image, expanded_mask,
                                filters=32, kernel_size=3, downsampling=1, reps=2) #
256x256
    conv2, mask2 = conv_bn_relu(conv1, mask1,
                                filters=64, kernel_size=5, downsampling=4, reps=2) #
64x64
    conv3, mask3 = conv_bn_relu(conv2, mask2,
                                filters=128, kernel_size=3, downsampling=2, reps=2)
# 32x32
    conv4, mask4 = conv_bn_relu(conv3, mask3,
                                filters=256, kernel_size=3, downsampling=2, reps=2)
# 16x16
    conv5, mask5 = conv_bn_relu(conv4, mask4,
                                filters=512, kernel_size=3, downsampling=2, reps=2)
# 8x8
    ## Decoder
    img, mask = conv_bn_relu(conv5, conv5,
                             filters=256, kernel_size=3, upsampling=2, reps=2,
                             concat_img=conv4, concat_mask=mask4) # 16x16
    img, mask = conv_bn_relu(img, mask,
                             filters=128, kernel_size=3, upsampling=2, reps=2,
                             concat_img=conv3, concat_mask=mask3) # 32x32
    img, mask = conv_bn_relu(img, mask,
                             filters=64, kernel_size=3, upsampling=2, reps=2,
                             concat_img=conv2, concat_mask=mask2) # 64x64
    img, mask = conv_bn_relu(img, mask,
                             filters=32, kernel_size=5, upsampling=4, reps=2,
                             concat_img=conv1, concat_mask=mask1) # 256x256
    img, mask = conv_bn_relu(img, mask,
                             filters=3, kernel_size=3, upsampling=1, reps=1,
act="tanh")
    ## 以下損失関数のレイヤーとかで略
```

詳細なコードはこちらを参照。

https://github.com/koshian2/DeepCreamPyBook/blob/master/Oppai/oppai_main.py

256×256の入力とし、最初だけ4倍のダウンサンプリングを入れ、あとは2倍のダウンサンプリング＋チャンネル2倍。深さを確保するため、reps分「P-Conv→BN→ReLU」を繰り返した。出力層はtanhとしている。カラースケールは、画像の出力は$[-1, 1]$のTensorFlowのスケール、VGGの計算はCaffeのスケールを使っている。U-NetとVGGで別のカラースケールを使っているのは、**損失関数のNan対策**である（後述）。

損失関数のレイヤーを入れるのはP-Convの章で見たとおりでほとんど変わらない。

なお、出力層に入力層をConcatするということもできる（例えば、入力層＋U-Netのtanhを2倍して入れると、U-Netは入力画像の差分を学習するだけのネットワークとなる）。しかし、この場合は、モザイクがかかった入力画像のPSNRは30以上もあり、U-Netによる復元画像がせいぜいPSNR20前半なので、損失関数的には入力画像のほうを選んでしまう。これはU-Net側に勾配が伝わらないということを意味し、U-Netはなにもせず、入力画像のコピーだけをする、したがって**モザイクには一切変化**がない出力結果となる。これはちょっとまずい。

しかし、入力画像の欠損度合い（PSNR）が変われば、例えば、モザイクではなく謎の光のような白抜きになったり、モザイクももっと粗いモザイクになったりして、入力画像のPSNRがもっと下がれば、この手法は収束速度が格段に速くなるので意味がある。こういったケースでは、入力層と出力層をつないだ際に、特に偏った最適化がされるということはわからなかった。ただし、U-Netはより質の良いSkip-Connectionのほうに「怠けたがる」ので、入力と出力をSkipで繋ぐ場合は**入力画像の粗さ**に注意しよう。

4. Nan対策

地味に沼だったのは損失関数のNan対策だ。TPUで訓練してたらずっとNanが出ていて特定するのが非常に大変だった（これだけで1日かかった）。なんとか奇跡的に特定してみたら、以下の3要素が複合的に絡んだ極めて面倒な案件だった。

1. **カラースケールによる問題**。U-Netを値域が-150～150ぐらいのCaffeのカラースケールだと、深いネットワークを作ったときに、値か勾配が爆発して、局所的に無限大ができてしまう。これが損失関数のNanを引き起こしていると思われる。
2. **Style Lossのグラム行列の問題**。これは 1 とも関連するが、256×256のような大きい解像度でStyle Lossのグラム行列を計算すると、(65536×65536)の行列の演算をしなければならない。これはいくらTPUでもメモリが尽きるし（Socket ClosedによるTPUのサイレントクラッシュが何度もあった)、なにせ65536個の掛け算、足し算をしているうちに、どこかでオーバーorアンダーフローを起こす。これが損失関数のNanを生み出しているのである。
   このグラム行列の問題は、カラースケールをTensorFlowにすると改善する。例えばCaffeのスケールだとAveragePoolingなどで数倍にダウンサンプリングしてから、グラム行列を計算してもまだNanが出るのに（入力画像が320×240で試したときに頻発した）、TensorFlowのカラースケールだとほぼ起こらない。
3. **解像度とネットワークの深さの問題**。これは一般的に疑いやすい問題だが、根底の原因が1,2にあるので、解像度を落とすことやネットワークを浅くすることが、Nanの直接的解決法かというと必ずしもそうではない。CNNでは通常、解像度や深さが原因でNanが出るということはない。しかし、1,2が改善されるため、解像度やネットワークを落としても偶然Nanが消えることもある。そうであっても、解像度や深さは直接は寄与してなさそうである。この切り分けが難しい。

そのため、U-NetではTensorFlowのカラースケール、VGGではCaffeのカラースケールを使う。

5. 損失関数の再チューニング

P-Convでは以下の損失関数を使っていた。これは論文に掲載されていた設定だ。

$$L = L_{valid} + 6L_{hole} + 0.05L_{perceptual} + 120(L_{style,out} + L_{style,comp}) + 0.1L_{tv} \tag{1}$$

これはU-NetとVGGでカラースケールが一緒なら使える。しかし、2つのネットワークのカラースケールが違っている状態でこの設定を使うと、Style Lossが効きすぎて元の画像の色調が大きく変わってしまうという現象が確認された。損失関数の再チューニングが必要だ。

$$L = 10L_{valid} + 60L_{hole} + 0.005L_{perceptual} + 1(L_{style,out} + L_{style,comp}) + 0.1L_{tv} \tag{2}$$

30ケース以上試したところ、このあたりのパラメーター設定がよいのがわかった。Style-Loss以外はあまり大した差はない。Style Lossを大きくしすぎるとあからさまに悪くなる。この式(2)を損失関数として利用した。

実行

訓練

リポジトリはこちら。**TPU環境**を使う。

https://github.com/koshian2/DeepCreamPyBook/tree/master/Oppai

Google Driveをマウント

```
from google.colab import drive
drive.mount("gdrive")
```

必要なファイルのダウンロード

```
!cp gdrive/My\ Drive/oppai/oppai_dataset.zip oppai_dataset.zip
!unzip oppai_dataset.zip > /dev/null
!apt install subversion > /dev/null
!svn export https://github.com/koshian2/DeepCreamPyBook/trunk/Oppai/libs --force
```

あとは上のリポジトリから、「**oppai_main.py**」をコピペして実行。

推論

「oppai_main.py」の中にあるplot_results()を使う。ただし、「oppai_train.hdf5」はコピーやアップロードなどで存在しているものとする。

おっぱいのモザイク解除結果

難しい話はさておいて、モザイク解除結果はどの程度になったか確認してみよう。画像は[*4]

左上がモザイクありの画像、右下がモザイクかける前のGround Truthである。右上がU-Netの推論結果であるが、先程説明した通りの着衣の状態での胸のモザイク消しはデータ数が足りないので、基本的にぼやける。そこで、**モザイクありの画像とU-Netの推論結果をアルファブレンディングして、モザイクを平滑化したような画像（左下）を解除結果**とした。真の画像までは鮮明にならなかったが、モザイクのゴツゴツした表情がなめらかになっている。より良いおっぱいとなった。

基本的に単色のほうがうまく行きやすく、複数の色からなるほど厳しくなっていった。画像は[*5]

Mosaic - Decensored - Blend - Ground Truth
(https://www.pixiv.net/member_illust.php?mode=medium&illust_id=72349625)

このケースの左下はかなり自然だ。単色のほうがうまくいきやすいのなら、**着衣より生おっぱいのモザイク消しのほうがうまくいきやすい**のは想像にかたくない。

いくつか面白いケースを紹介しよう。**どうもこのネットワーク、服を脱がせたがる**ような傾向がある。例えばこのケース。画像は[*6]

Mosaic - Decensored - Blend - Ground Truth
(https://www.pixiv.net/member_illust.php?mode=medium&illust_id=73613537)

右上がかなり大変なことになっている。なんか肌色っぽくなっているが多分肌色のパジャマに変換されたのだろう。そして髪の毛の先がピンク色の突起のようになっているようにも見えるが、もともとこのあたりに手があったからきっと手に違いない。左下のブレンディングしている画像は、パジャマの布地が薄くなったように変換されているが、ネットワークの推論結果が肌色強めだとこういうことが起こるのである。**モザイクを解除しないで、服を解除しようとしてどうするねん。**

これもなかなかすごい。画像は[*7]より。右上の推論結果はきっと肌色のブラウスの谷間にネクタイを挟んだのだろう。R-18のアレではない。ブレンディング画像は、透けブラのような雰囲気になっているのも面白い。

はじめの2例だけ見れば、「ニューラルネットワークを使わなくても、例えばガウシアンブラーで元のモザイクにぼかしを入れて、アルファブレンディングすればいいじゃん」と思うかもしれないが、ニューラルネットワークのケースでは今見たように予想外の面白い結果が出てくるので、やはり**モザイク消しにはニューラルネットワーク**ではないかなと思う。もちろん、面白いだけでなくデータを確保をできれば精度は出る。ニューラルネットワークにはポロリの夢がある。最新の技術であるが、日本の古き良き文化を踏襲した素晴らしい技術だ。

最後に一例だけ。この右上の推定結果はとてもおもしろい。**服のおっぱいの部分の白い箇所だけ綺麗さっぱり消えているのだ**。もともとサンタ服であったので、残った赤い部分でギリギリ隠れているのが頼もしい。

これはとてもニューラルネットワーク的な挙動だ。モザイクがついているか、服の色がどうかという狭い領域の特徴量だけでなく、**そこが胸かどうかという大域特徴量に注目している**のである。つまり、ニューラルネットワークは「おっぱいのモザイクを消しなさい」という概念を獲得しているのだ。

しかし、右上の状態、どうやって服の形状を維持しているのだろうか。見えない絆創膏でもあって貼り付けているのだろうか。

We need GANs

いくら面白い出力を出すと言っても、服の上からのおっぱいのモザイクを鮮明に取り除くというのは、本書で見たようなP-Conv U-Netでは難しかった。これは服や装飾品のバラエティを満たすだけのデータがない他にもうひとつ原因がある。それはP-Convのような**損失関数では限界がある**ということだ。

もちろんP-Convの損失関数は多彩なことができて便利だ。しかし、われわれが見ているのは、Style LossやContent Lossといった物理レベルの量ではなく、「出てきたものがもっともらしいか」「出てきたものが本物っぽいか」という**メタな指標**である。出てきた画像に対して、「これは違うよな〜」というのもメタな指標である。GANはGeneratorとDiscriminatorが互いにゲームをして性能を向上させていくし、Discriminatorのやっていることが「本物か偽物かを見分ける」ということだから、GANはメタな指標を損失関数としている。

考えてみれば、損失関数が固定というのも表現力には限界があるように思える。おっぱいを見るか、顔を見るかという部位の違い、マスク領域を見るか非マスク領域を見るかという領域の違い、写真のモザイクを消すのかイラストのモザイクを消すのかというドメインの違い、そして学習がどの程度進んでいるのかというフェーズの違い——これらが一個の損失関数で表されるというのは到底考えられない。そう、ロジスティック回帰からニューラルネットワークに進化したような**パラダイムシフトが必要**なのである。

よりテクニカルには、P-Convで使ったような**L1距離による損失関数というのはボケやすい**というのが指摘されている。まさに、今回のおっぱいモザイクで見たとおりだ。pix2pixという論文[*9]からの図である。

真ん中のL1というのがL1距離だ。その横にあるのはGANを使った場合の出力で、こんなに差が出るのだ。GANのパワー、やはりほしい。

これに気づいたのが締め切りの数日前なのが本当に残念だが、GANを使ったらもっと鮮明にモザイク解除ができたかと考えるとなかなかに悔しい結果となってしまった。

こしあん王国〜その後〜

そういえば、テレビでおっぱいにモザイク規制を入れるという、おっぱい規制を発表したこしあん王国はどうなっただろうか。その後を見ていこう。

あなたの功績により、そこまで鮮明さは高くはないものの、モザイクの解除装置のプロトタイプの開発に成功した！　規制施行当初は、予想通りモザイクに対する落胆の声もあった。しかし、あなたが作ったモザイク解除装置が、たまにとても興味深い解除結果を出すため、あなたが恐れたほど国民の反発は強くはなかった。

試験期間が終わるころには、モザイクの解除装置を使うと、モザイクなしでは普段見ることができないポロリ（らしきもの）が見れるというのは、国民の周知の事実となった。一部ではモザイクの存続すら望む声が出ていた。「モザイクでポロリが見れるなんてなんのためにモザイクかけたのかよくわからんな」「この解除装置を作った人は天才だな、一体誰なんだ」「これを見越して規制を実施した女王はやはりすごい」などなど。本来意図した形ではなかったが、女王の聖なる威光は示されたのである！

しかし、一人不機嫌な者がいた。そう、アン女王本人である。「はぁ？　モザイクのほうが良いなどと下賤の者は何を考えているのじゃ。わらわだけ見ていればいいのに」と八つ当たりされる日々。

それで終わることはなかった。ほどなく、ここぞとばかりに商機を見出したものが「胸にモザイク模様の入った服」というのを売り出した。モザイクへの不満と一部の熱狂的な支持が合わさり、この服はまたたく間にブレイクした。この規制のきっかけとなった、あなたがテレビで見ていたナイスバディな美少女アイドルが、この服を着たのもブームを推し進めた。

これを見てアン女王がついに陥落した。胸にモザイク模様の入った服が可愛いからという理由で、おねだりをし始めたのである。

ほどなくして、おっぱい規制は期間満了により終了し、その後復活することはなかった。ここでのモザイク解除技術の記録は、その後の王国の情報技術や軍事技術に遺憾なく発揮された。あなたがこの不思議なモザイク解除装置の開発に携わったことは、メイド長以下、宮廷のごく少数の人間のみぞ知る事実となった。

謝辞

この章ではPixivの偉大なイラストを、機械学習のしょうもないタスクの学習素材として活用させていただいた。この章はイラストレーターの方々による、素晴らしいおっぱいがなければ成り立たなかった。

モザイクの解除という視点から見たが、**美しいおっぱいの描画というのはAIを使ってもまだまだ難しい**というのがよくわかった。人工知能は将棋の名人を破っても、おっぱいにはまだまだ負けてしまうのである。それだけイラストレーターのやっていることは高度なことであるし、そのことに深く尊敬の念を表したい。偉大なるおっぱいをどうもありがとう。

問題（強い人向け）

もうあえて問題という問題というものはないが、どちらかというと私の無念を晴らしてほしいので、以下のアイディアを元にどれかやってみると成功するかもしれない。全て強い人向けの内容である。

1. モザイク解除自体がリサイズ、つまりPoolingとUpsamplingを噛ませているのだから、モザイク解除の原理的な計算はDeconvolution（Conv2DTranspose）に似ているように思われる。このレイヤーを使ってP-Conv U-Netを作るとうまくいくのだろうか？
2. 深いU-Netでは勾配が伝わりにくいという現象がある。ResNetのようなSkipConnectionをP-Convに入れてみよう。
3. P-Conv U-Netはそのままで、訓練方式をGANを使ったpix2pixに変えてみよう。

参考文献

1. 【SSSS.GRIDMAN】「新条アカネ」/「富岡二郎」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=72656796
2. 【ホムラ】「焔」/「Ormille」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=72809999
3. 【プロジェクト東京ドールズ】「【東京ドールズ】水着[2017]衣装・まとめ」/「惰猫」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=72517304
4. 【アネモネ(花騎士)】「☘」/「VelmAr」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=72918799
5. 【イラスト】「クリモネ2018」/「キムキム◇PFLSなう」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=72349625

6. 【間桐桜】「せんぱい？」/「橙」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=73613537
7. 【ブレイブルー】「Esちゃん」/「たかなしＡ」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=71798453
8. 【プリンセスコネクト!Re:Dive】「くーりすますがことしもやぁてくる」/「アクアス」のイラスト [pixiv] https://www.pixiv.net/member_illust.php?mode=medium&illust_id=72197185
9. P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. CVPR 2017. 2016. https://arxiv.org/abs/1611.07004

裏話

機会があって病院に行くことがあったのだが、夜な夜なノートパソコン広げて（許可は取ってある）、おっぱいデータセットのアノテーション作業をしていたことがある。おそらくこんなの一生ないだろうなと思う。おっぱいデータセットはそういった裏話があるのでぜひ皆さんも使って欲しい。

2019年4月　Shikoan’s ML Blogにて公開