ISSN 2186-5647

P-29

Fast GAN with Mixing Regularization

日大生産工　成田　輝　　日大生産工　山内　ゆかり

1. まえがき

ニューラルネットワークで画像の識別を行う際、人が学習データに写っているものに慎重にラベル付けを行う必要があり、膨大な作業量を要する。イアングッドファエローはこの問題を解決するため、単一のニューラルネットワークではなく２つのネットワークを競合させるGANを提案した。GANは膨大な作業量を大幅に削減するだけでなく、いままでにない高度な識別能力を有したネットワークと学習データに非常に似た出力データを作成するネットワークの作成にも成功した。しかし、学習難易度の高さからネットワークが上手く学習しないことが起きる。

本研究では、GANにおける上記の問題を解決するため、Fast GAN[1] + Mixing regulationを提案する。

2. 従来研究

Fast GAN[1]の主なアーキテクチャとして、skip-layer excitation と self-supervised discriminatorが挙げられる。

2.1 Generator

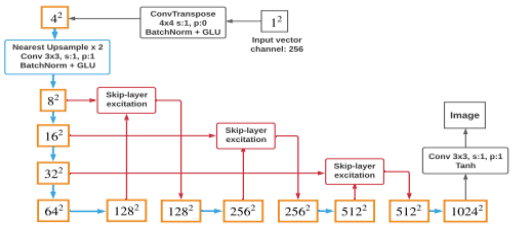

Figure 1 Generator

まず、黒枠で入力ベクトルチャンネルに対して転置畳み込み＋バッチ正規化を行い、活性化関数には GLU を使う。GLU 関数とは特徴マップを 2 分割し、片方に Sigmoid を通して積を取る関数である。次に青枠で Upsample+畳み込みを行い、サイズを拡大する。ここでもバッチ正規化、GLU を扱う。そして次に行う赤枠の処理は Skip-Layer Excitation 呼ばれ、Fast GAN の肝となるモジュールの一つである。

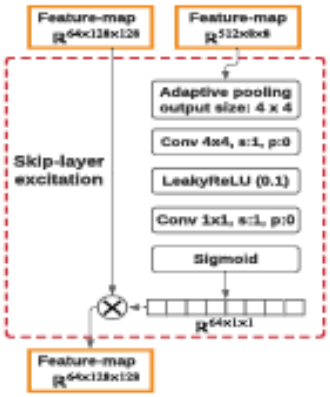

Figure 2 Skip-Layer Excitation

サイズの大きい特徴マップと小さい特徴マップの二つを入力として受け取り、二つを合成して返すモジュールである。合成後は大きい特徴マップと同じサイズになる。モジュールの中身として、まず 4×4 のサイズになるよう適応平均値プーリング、次に 4×4 の畳み込みを行い、活性化関数として LeakyReLU を通す。次に 1×1 の畳み込みをしたら、Sigmoid 関数を通し、一次元のテンソルにする。最後に大きい特徴マップとテンソルを掛け合わせて合成した特徴マップを出力する。

2.2 self- supervised Discriminator

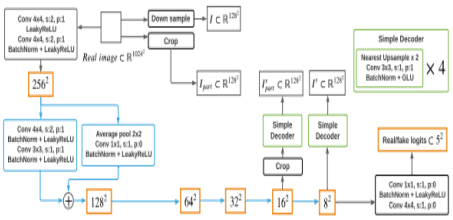

Figure 3
self- supervised Discriminator

この Discriminator の特徴として、16×16(f1)と 8×8(f2)の二つの特徴マップのために Simple Decoder が採用されている。f1 は 8×8 サイズにランダムサンプリングされてから Decoder に入力として渡される。Decoder の処理は右上の緑枠に書いてあり、Upsample+畳み込み

Fast GAN with Mixing Regularization

Akira NARITA and Yukari YAMAUCHI

+BN+GLU を 4 回繰り返す。サンプリングされた f1 を入力とした Decoder の出力をI'_{part}、f2 を入力とした Decoder の出力をI'とする。そして Real image から 1/8 サイズでランダムサンプリングしたものをI_{part}、単純に縮小してリサイズしたものをIとする。Decoder の誤差関数を表す式は以下になる。

$$Lrecons = E_{f\sim Dencode(x),x\sim Ireal}[||G(f) - T(x)||] \quad (1)$$

I'_{part}はI'、I_{part}はIとの差分を取ることで誤差関数を求めることができる。このように Discriminator を AutoEncoder 的構造にすることで、Discriminator がより包括的に特徴を学習するようになる。

2.3 Generator と Discriminator の誤差関数

$$LD = -E_{x\sim Ireal}[min(0,-1 + D(x))] - E_{\hat{x}\sim G(z)}[min(0,-1 - D(\hat{x})] + Lrecons \quad (2)$$

(2)は Discriminator の誤差関数を表しており、Hinge loss を使用している。デコーダの誤差である$Lrecons$ (1)は識別機に本物画像を入れている場合のみ加える。

$$LG = -E_{z\sim N}[D(G(z))] \quad (3)$$

(3)は Generator の誤差関数を表している。

3. 提案手法

Figure 4 Skip-Layer Excitation による Style Mixing

低時間の学習において、Fast GAN は色のスタイル変換には優れているが、Style GAN のように構成パーツの交換には優れていない。そこで各層の style の影響を他の層にも局在化できるよう Mixing Regulation を組み合わせた Fast GAN を提案する。

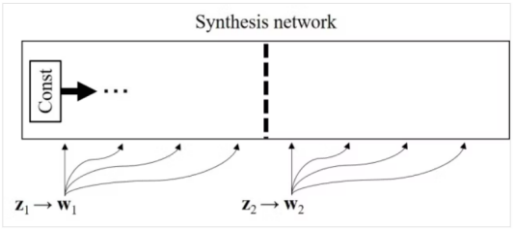

Figure 5 Mixing Regularization

Mixing Regularization とは、合成ネットワーク内のレイヤーをランダムに選択し、そこを機に潜在変数 z1、z1 からマッピングされる w1 を z2、z2 からマッピングされる w2 に切り替える手法である。Style GAN ではこの正則化手法を取り入れることにより、style の影響を大きく変化させることに成功している。

4. 実験および検討

今回の実験において、データセットはFast GANでも用いられていたArt Paintings(1000)、Pokemon(800)、Skull(100)を使用する。プログラミングは、Visual Studio Code上でPythonを使って行う。また定量評価指標として、生成画像と本物画像の平均と共分散を比較することでどれだけ本物画像に似ているかを表すFID値を扱う。次に実験方法として、従来のFast GANと提案手法のMixing Regularization + Fast GANを用いてStyle Mixingで画像生成を行う。そのときのデータセットのクラス数は2k、5k、10kごとに、学習時間は10h、20h、30hごとに学習を行う。

5. まとめ

各層のstyleの影響を他の層にも局在化できるようにするためStyle GAN内で使われているMixing RegularizationをFast GANに組み込むことで、Fast GANの処理速度を保持したまま構成パーツの交換も可能になるのではと考えている。

参考文献

[1] Bingchen Liul, Yizhe Zhu, Kunpeng Song, Ahmed Elgammal:TOWARDS FASTER AND STABILIZED GAN TRAINING FOR HIGH-FIDELITY FEW-SHOT IMAGE SYNTHESIS