[1] GANとは？

敵対的生成ネットワーク（GAN: Generative Adversarial Nets）論文 NeurIPS 2014

生成モデルの一種

生成器と識別器が交互に学習を進めて、本物と見分けの付かない出力が得られる

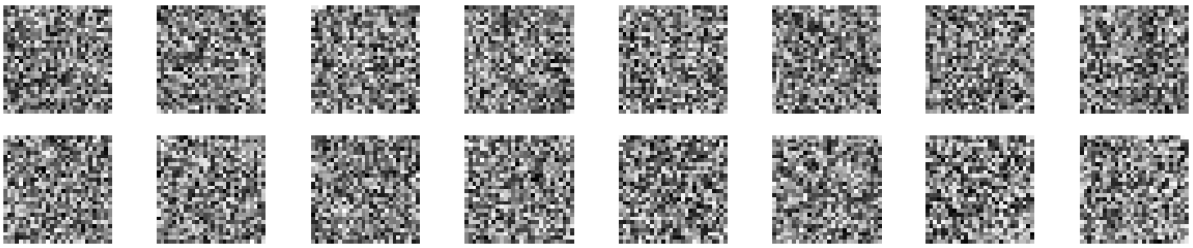
著者の講演会：Introduction to GANs, NIPS 2016 | Ian Goodfellow, OpenAI

GANの派生モデル

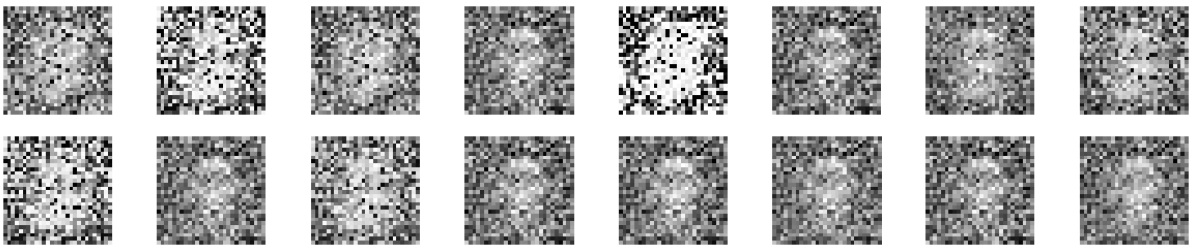
- DCGAN(Deep Convolutional GAN)
  畳み込み層を含んだモデル
- WGAN(Wasserstein GAN)
  JSダイバージェンスの代わりに、Wasserstein距離を用いて学習を安定させたモデル
- 他にもたくさん
  気になる方は → 主要なGAN研究の歴史（2019年11月現在）

[2] GANを動かしてみる

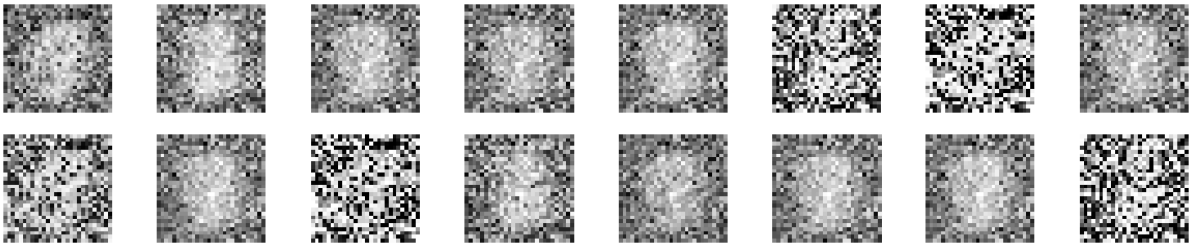
実験 ① GAN.ipynb：シンプルなMLPでの結果
z.shape = randn(BATCH_SIZE, 32)
[EPOCH 00]



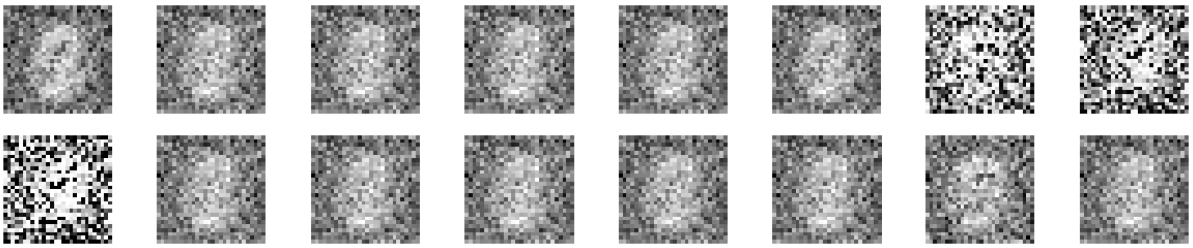
[EPOCH 01]



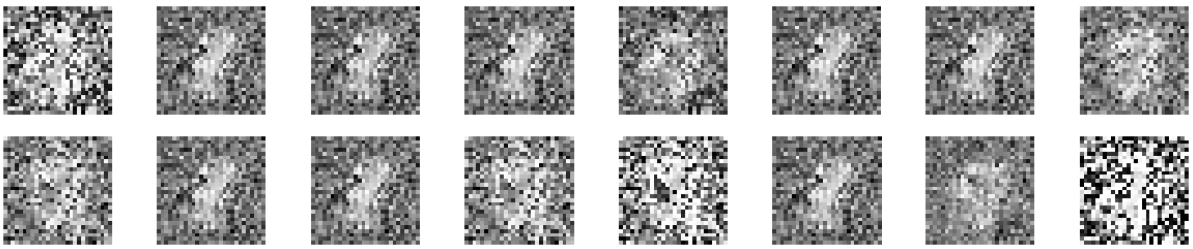
[EPOCH 02]



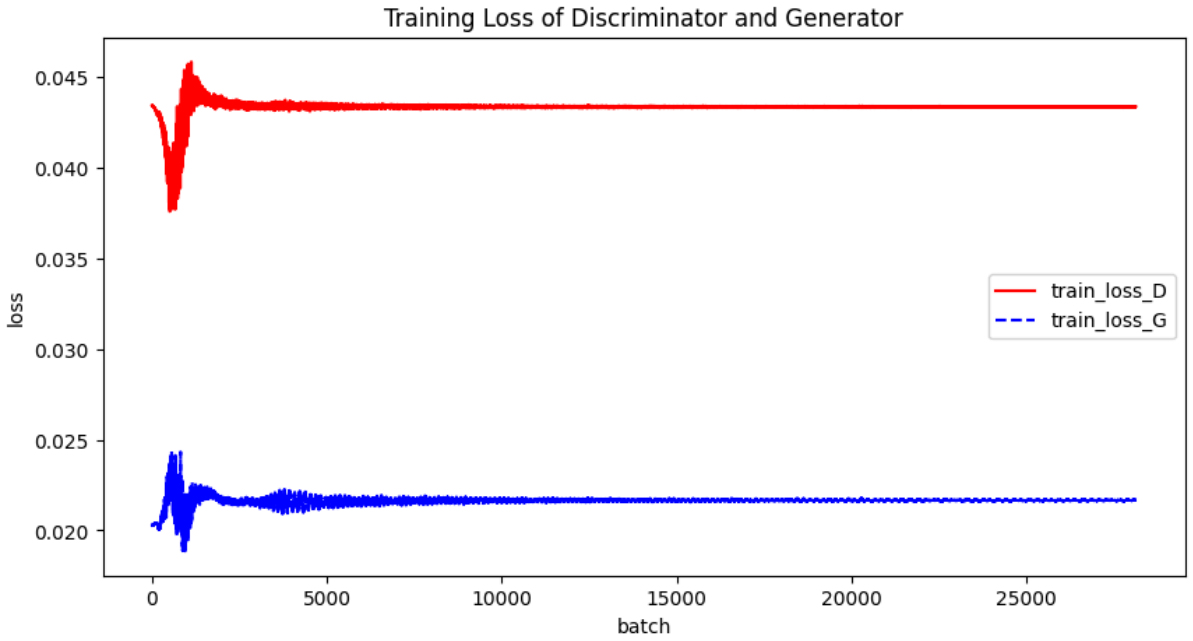
[EPOCH 10]



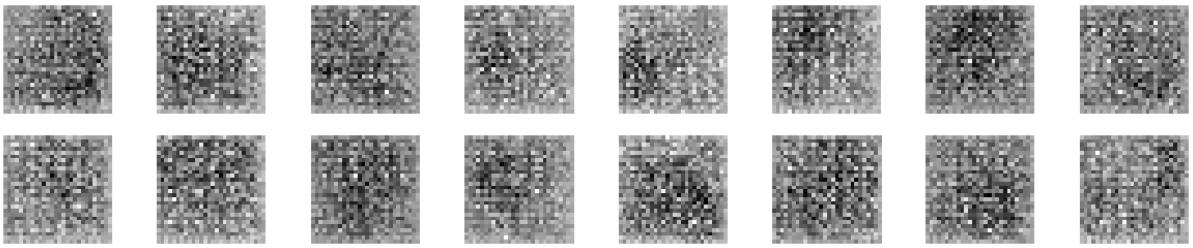
[EPOCH 15]



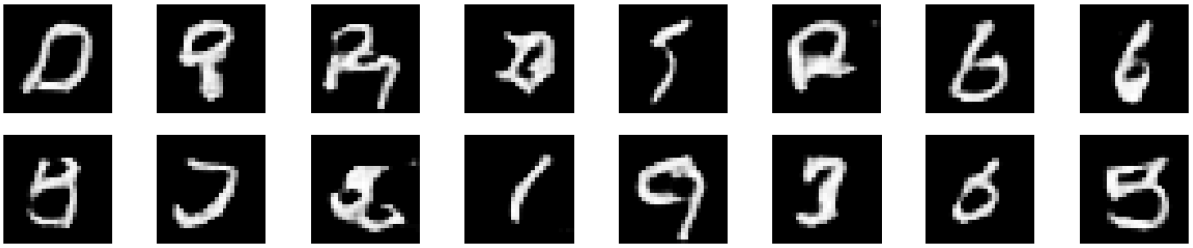
3 EPOCH分の学習にかかった時間：約10分（T4 GPU ハイメモリ）
モード崩壊が確認できる



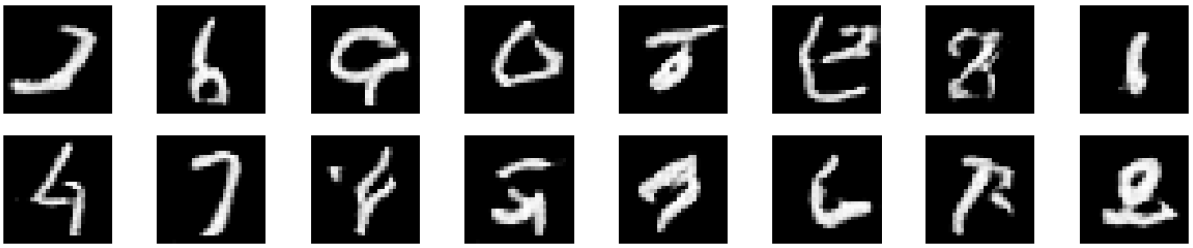
実験 ② DCGAN.ipynb : 畳み込み層を用いたGANの結果
z.shape = randn(BATCH_SIZE, 1, 7, 7)
[EPOCH 00]



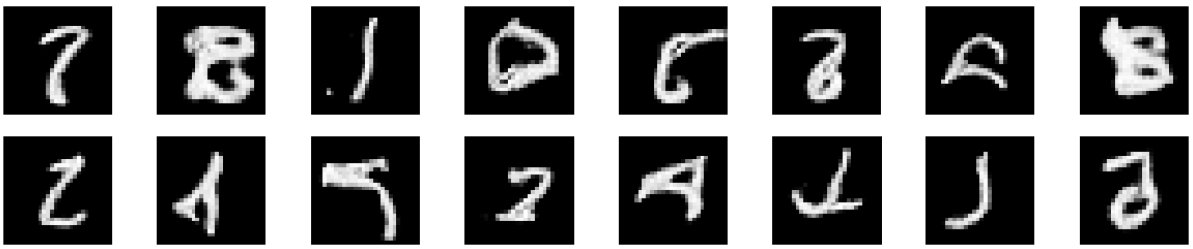
[EPOCH 01]



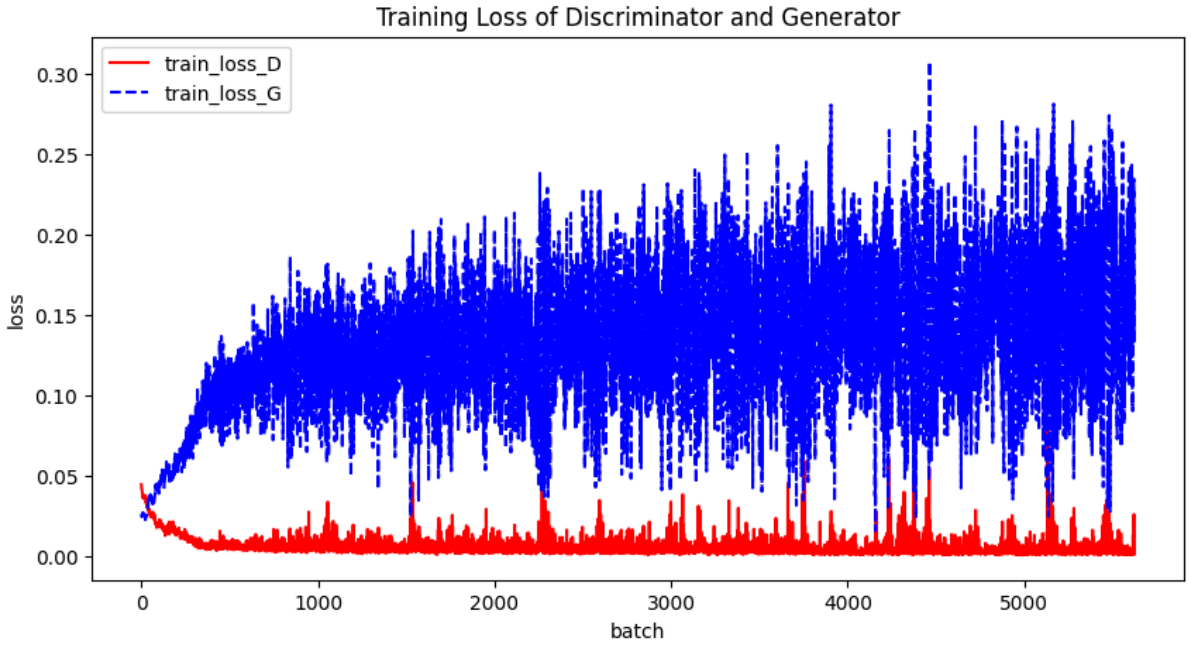
[EPOCH 02]



[EPOCH 03]



3 EPOCH分の学習にかかった時間 : 約8分（T4 GPU ハイメモリ）
※ もっと時間かけて学習させれば、綺麗な画像ができるみたい



「参考」

【Pytorch】MNISTのGAN（敵対的生成ネットワーク）を実装する #Python3 - Qiita

（コード変更部分あり）

「コラム」

今回の実験①, ②では、潜在表現の次元数とサンプリング対象の確率密度関数をテキトウに設定しています。正規分布(torch.randn)ではなく、一様分布(torch.rand)にしてみるとか、変更して出力結果が良くなるか調べても良いかも？

生成モデルには、拡散モデル（Diffusion Model）があり、GANのモード崩壊を解決しているらしい。