Applying DCGAN on anime images

PLXIV
Universitat Politècnica de Barcelona.
2019

Block 1:
Presentation

Project Scope and motivation.

- Build an anime avatar generator using a simple GAN’s.
- Learn and take my first steps towards GAN’s.
- Seemed kinda fun.

State of the art and context on anime projects (Topics)

Main topics related with anime:

- Anime generators
- Increase resolution
- Paint images
- Decensoring images
- Human to cartoons.

State of the art and context on anime projects (Papers)

- [2018,2019] Generate anime (avatar and full body) images. X
- [2017] Paint anime images using style transfer AC-GAN
- [2018] Decensor anime using PEPSI
- [2016] Increasing the resolution of the images.
- [2019] Transformation of human image into a cartoon image using CycleGAN.

https://github.com/deeppomf/DeepLearningAnimePapers

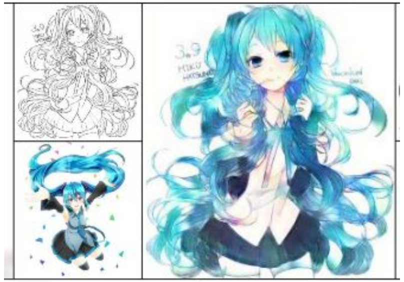
State of the art and context on anime projects (Examples)

Anime generator

Decensoring

Human to anime

Style transfer



Block 2:
Implementation

Steps followed (Planning)

- Obtain meaningful dataset
- Preprocess the dataset
- Choose a model
- Tune the parameters
- Train
- Evaluate and understand

Dataset (Source)

1. Danbooru is a web page where users can updates their creations.
2. It is fairly well tagged and contains millions of images.
3. It provides an api which allows to download the desired images.
4. 3M. of images where downloaded from this source.

https://danbooru.donmai.us/

Danbooru

Dataset Preprocessing (crop_faces.py, upscale.py)

The steps performed to preprocess the images are the following ones:

- Obtain the face animes using *nagonamis* face recognition module (500.000 anime faces were obtained).
- Remove colourless faces.
- Resize images to 64x64 resolution.
- Convert all values from 0 to 255 to 0 to 1.

Generative adversarial network architecture (nn_models.py)

```
Model: "Discriminator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 1, 1, 100)         0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 4, 4, 512)         819712
_________________________________________________________________
batch_normalization_1 (Batch (None, 4, 4, 512)         2048
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 4, 4, 512)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 512)         0
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 8, 8, 256)         2097408
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 8, 8, 256)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 16, 16, 128)       524416
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_transpose_4 (Conv2DTr (None, 32, 32, 64)        131136
_________________________________________________________________
batch_normalization_4 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
batch_normalization_5 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_transpose_5 (Conv2DTr (None, 64, 64, 3)         3075
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 3)         0
=================================================================
Total params: 3,616,771
Trainable params: 3,614,723
Non-trainable params: 2,048
```

```
_________________________________________________________________
Model: "Generator"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        3136
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       131200
_________________________________________________________________
batch_normalization_6 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
dropout_6 (Dropout)          (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 256)         524544
_________________________________________________________________
batch_normalization_7 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 8, 8, 256)         0
_________________________________________________________________
dropout_7 (Dropout)          (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 4, 4, 512)         2097664
_________________________________________________________________
batch_normalization_8 (Batch (None, 4, 4, 512)         2048
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 4, 4, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 8193
_________________________________________________________________
activation_2 (Activation)    (None, 1)                 0
=================================================================
Total params: 2,768,321
Trainable params: 2,766,529
Non-trainable params: 1,792
```

Training hyperparameters (train.py)

Parameters of the training set:

- Optimizer: Adagram
- Learning rate: very low, slightly bigger on the generator than the discriminator (0.0002, 0.00015)
- Loss: binari_crossentropy
- Epochs: 2000~
- batch size: 256
- Training time: 1h.

Results (Generator)

After one 2000~ epochs of training

Block 3:
Evaluate results

Conclusions and knowledge

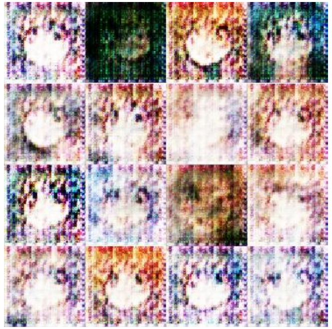
- Evaluate results
- Improvements
- Conclusions

Results (Generator)

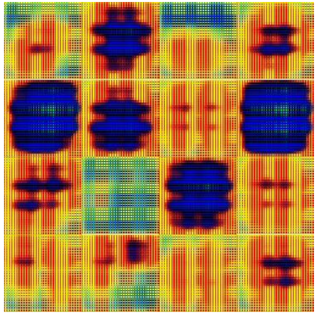
After one 2000~ epochs of training

Results on other configurations (Generator)

low momentum



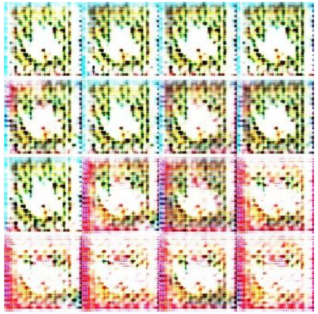
high lr



no_dr low momen

0.4 dropout

high dropout,high lr



no dropout

Loss plot

Improvements (Some possibilities)

- Play more time with the architecture
- Similarities inside the discriminator
- Increase data or select specific types of faces.
- Increase the training epochs
- One-side labeling smoothing
- Increase resolution of the images before resizing.
- Experience replay
- Change the cost functions
- Scale to -1 to 1 instead of 0 to 1 and use tahn.
- Use more complex and fitting GAN’s

Conclusions (Personal)

- Gan’s are really hard to train, it is hard to have a balance between the discriminator and the generator.
- It is hard to determine which parameters should be modified.
- Although itself was very fun it may me realize how complex problems could be.
- The “GAN world” is huge and it could be a field by itself.
- (Reserved question to the teacher)

Thanks for hearing
聞いてくれてありがとうございます

lilin.pau@gmail.com