

---

# MLDS HW3-1

TAs  
ntu.mldsta@gmail.com

---

# Update

- 5/8 : 更新Baseline Model

# Baseline Model (<sup>5</sup>/<sub>8</sub> Update)

- Generator

- input = (100, )
- Dense(128\*16\*16, 'relu')
- Reshape((16, 16, 128))
- Upsampling
- Conv2D(128, kernel = 4)
- Relu
- Upsampling
- Conv2D(64, kernel = 4)
- Relu
- Conv2D(3, kernel = 4)
- tanh

- Training

- Adam(lr = 0.0002, beta = 0.5)

- Discriminator

- input = (64, 64, 3)
- Conv2D(32, kernel = 4)
- Relu
- Conv2D(64, kernel = 4)
- ZeroPadding
- Relu
- Conv2D(128, kernel = 4)
- Relu
- Conv2D(256, kernel = 4)
- Relu
- Flatten
- Dense(1, sigmoid)

# Outline

- ❖ **Timeline**
- ❖ **Task Descriptions**
- ❖ **Model & Training tips**
- ❖ **Submission & Rules**
- ❖ **Q&A**

# Timeline

# Three Parts in HW3

- (3-1) Image Generation
- (3-2) Text-to-Image Generation
- (3-3) Style Transfer

# Schedule

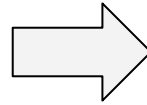
- 5/4 :
  - Release HW3-1
- 5/11 :
  - Release HW3-2
- 5/18:
  - Release HW3-3
- 6/8:
  - All HW3 due (including HW3-1, HW3-2, HW3-3)
- 上台分享

# Task Descriptions

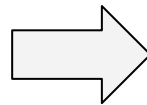


# HW3-1: Image Generation <sub>2/2</sub>

**Bird  
Generative Model**

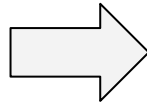


**Flower  
Generative Model**



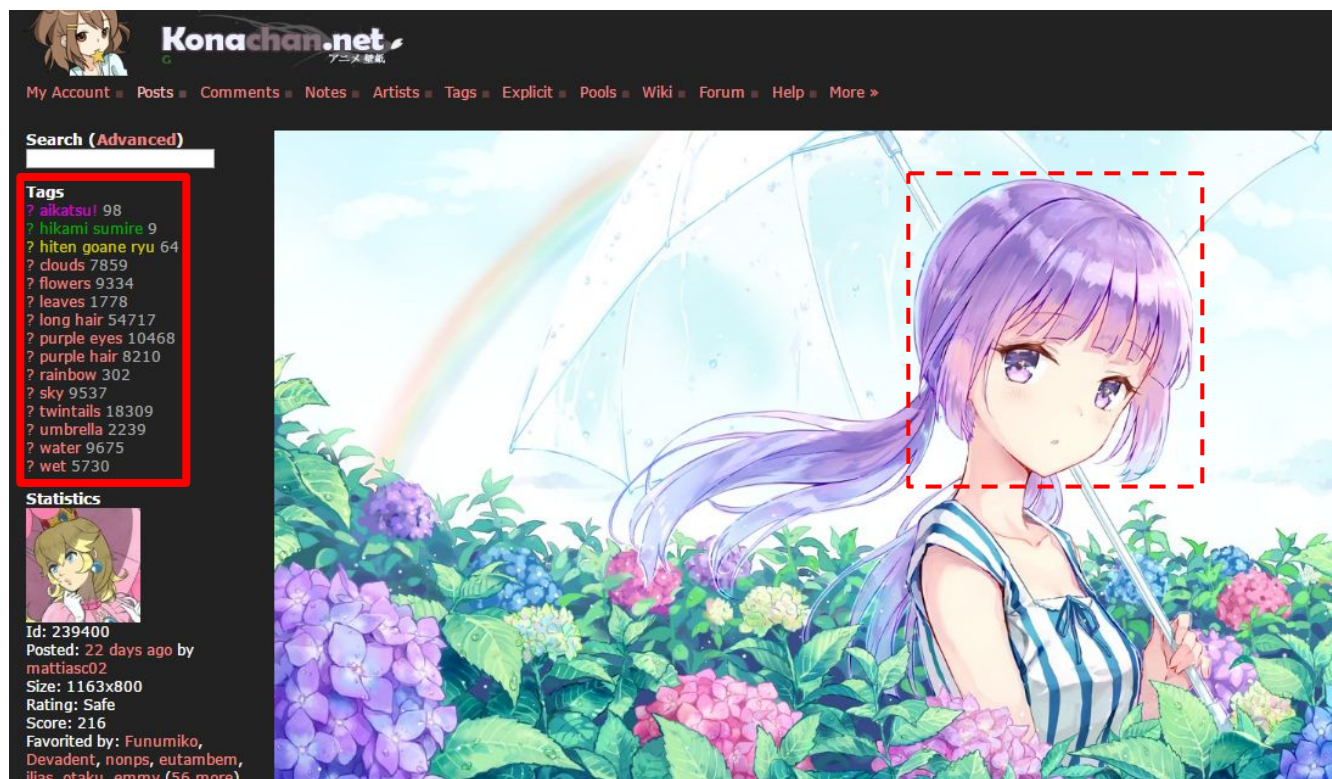
# HW3-1: Image Generation <sub>2/2</sub>

**Anime  
Generative Model**



# Data Collections <sup>1/2</sup>

- Anime dataset




[http://konachan.net/post/show/239400/aikatsu-clouds-flowers-hikami\\_sumire-hiten\\_goane\\_r](http://konachan.net/post/show/239400/aikatsu-clouds-flowers-hikami_sumire-hiten_goane_r)

感謝樊恩宇助教蒐集data

# Data Collections <sup>2/2</sup>

- Extra data

**MakeGirlsMoe** Home History Transition Help ▾



Generate

👍 +1

👎 -1

Share on Twitter

### Options

☐ Advanced Mode

Model

Camellia 256x256 Ver.171219 (9.9MB)

Hair Color

Random

Hair Style

Random

Eye Color

Random

Dark Skin

Off

Random

On

Blush

Off

Random

On

Smile

Off

Random

On

Open Mouth

Off

Random

On

Hat

Off

Random

On

Ribbon

Off

Random

On

Glasses

Off

Random

On

Style

Random

Noise

Random

Fixed

Current Noise

Noise Import/Export

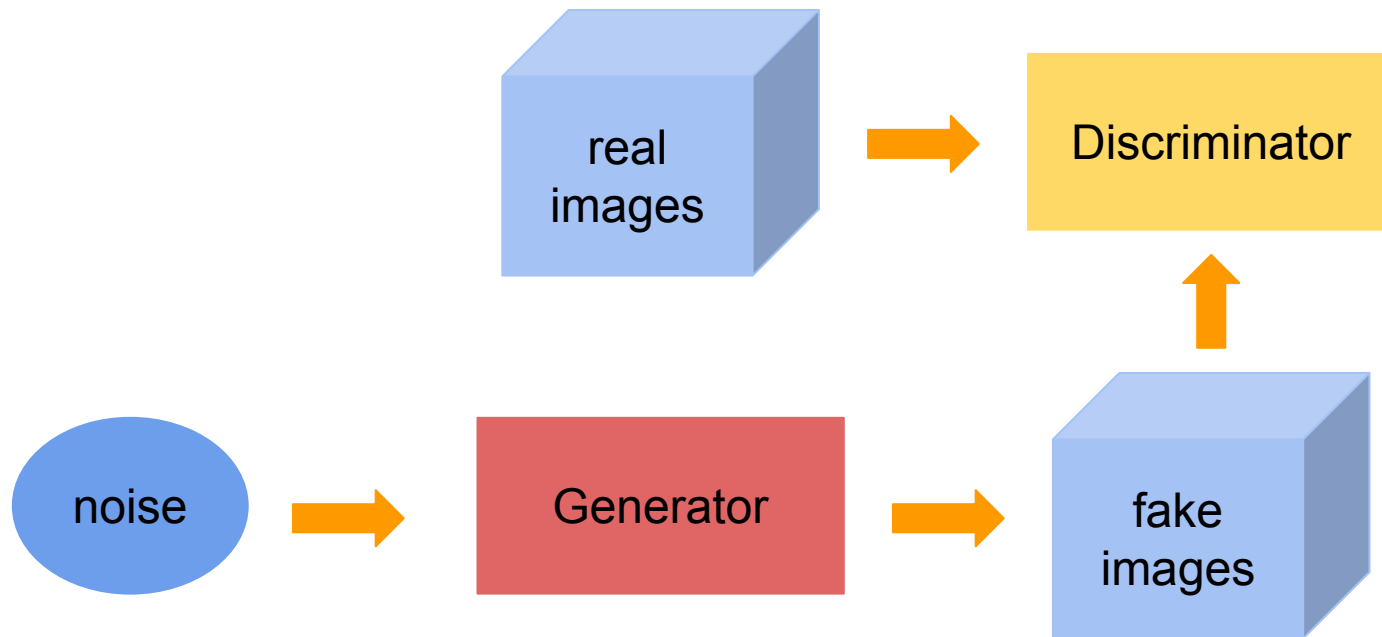
Import

Export

# Model & Training Tips

# GAN <sup>1/5</sup>

- Overview



# GAN 2/5

- Discriminator

- input: images (batch\_size, height, width, channels)
- output: scores (batch\_size,)
- architecture: CNN, DNN

- Generator

- input: noises (batch\_size, noise\_dim)
- output: images (batch\_size, height, width, channels)
- architecture: CNN, DNN
- use deconvolution (transpose convolution) layer in CNN

# GAN <sup>3/5</sup>

- Training procedure
  - repeat max\_iteration times
  - repeat d\_update times
  - discriminator = **update\_discriminator**(training\_data, generator)
  - repeat g\_update times
  - generator = **update\_generator**(discriminator)
- Testing procedure
  - noise = sample\_batch\_noise(batch\_size, noise\_dim)
  - output\_images = generator(noise)



# GAN 4/5

- Update discriminator

- `real_images = sample_batch_data(training_data, batch_size)`
- `noise = sample_batch_noise(batch_size, noise_dim)`
- `fake_images = generator(noise)`
- `real_predicts = discriminator(real_images)`
- `fake_predicts = discriminator(fake_images)`
- `d_loss = loss_d_fn(real_predicts, real_labels, fake_predicts, fake_labels)`
- `d_grad = gradients(d_loss, d_params)`
- `d_params = updates(d_params, d_grad)`
- `# do not` update the parameters of generator

- Update Generator

- `noise = sample_batch_noise(noise_dim, batch_size)`
- `fake_images = generator(noise)`
- `fake_predicts = discriminator(fake_images)`
- `g_loss = loss_g_fn( fake_predicts, real_labels )`
- `g_grad = gradients(g_loss, g_params)`
- `g_params = updates(g_params, g_grad)`
- `# do not` update the parameters of discriminator

# Wasserstein GAN <sup>1/3</sup>

The output of D is thus not probability anymore.  
The D loss turn to be a measure of distance.

$$L_D^{WGAN} = E[D(x)] - E[D(G(z))]$$

$$L_G^{WGAN} = E[D(G(z))]$$

$$W_D \leftarrow \text{clip\_by\_value}(W_D, -0.01, 0.01)$$

ref:<https://arxiv.org/abs/1701.07875>

# Wasserstein GAN 2/3

- In each training iteration: **No sigmoid for the output of D**

Learning  
D

Repeat  
k times

- Sample m examples  $\{x^1, x^2, \dots, x^m\}$  from data distribution  $P_{data}(x)$
- Sample m noise samples  $\{z^1, z^2, \dots, z^m\}$  from the prior  $P_{prior}(z)$
- Obtaining generated data  $\{\tilde{x}^1, \tilde{x}^2, \dots, \tilde{x}^m\}$ ,  $\tilde{x}^i = G(z^i)$
- Update discriminator parameters  $\theta_d$  to maximize
  - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m D(x^i) - \frac{1}{m} \sum_{i=1}^m D(\tilde{x}^i)$
  - $\theta_d \leftarrow \theta_d + \eta \nabla \tilde{V}(\theta_d)$  **Weight clipping**

Learning  
G

Only  
Once

- Sample another m noise samples  $\{z^1, z^2, \dots, z^m\}$  from the prior  $P_{prior}(z)$
- Update generator parameters  $\theta_g$  to minimize
  - $\tilde{V} = \frac{1}{m} \sum_{i=1}^m \log D(x^i) - \frac{1}{m} \sum_{i=1}^m D(G(z^i))$
  - $\theta_g \leftarrow \theta_g - \eta \nabla \tilde{V}(\theta_g)$

# Wasserstein GAN <sup>3/3</sup>

- Implementation Notes:
  - Do not apply sigmoid at the output of D
  - Clip the weight of D
  - Use RMSProp instead of Adam
  - Train more iteration of D (the paper use 5)

# Improved WGAN (WGAN-GP) <sup>1/2</sup>

Do not clip the weight of D but to add a new objective called “Gradient Penalty”.

$$L_D^{WGAN\_GP} = L_D^{WGAN} + \lambda E[(\|\nabla D(\alpha x + (1 - \alpha)G(z))\|_2 - 1)^2]$$

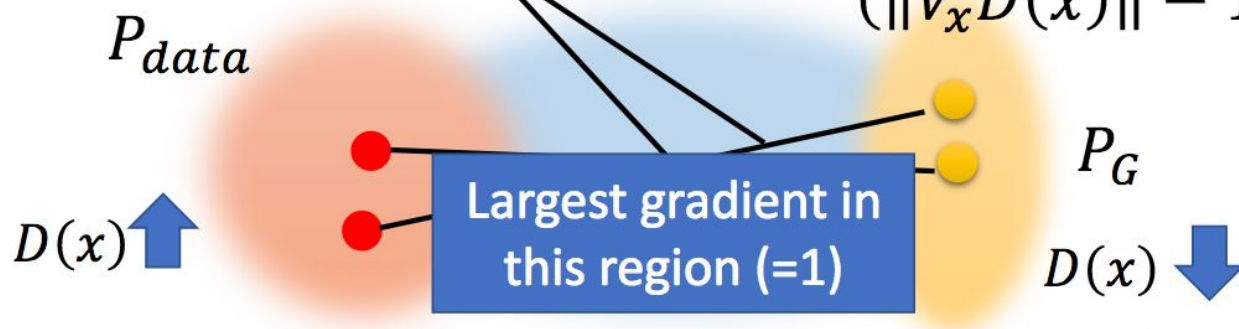
$$L_G^{WGAN\_GP} = L_G^{WGAN}$$

ref:<https://arxiv.org/pdf/1704.00028.pdf>

# Improved WGAN (WGAN-GP) <sup>2/2</sup>

$$W(P_{data}, P_G) \approx \max_D \{ E_{x \sim P_{data}} [D(x)] - E_{x \sim P_G} [D(x)] - \lambda E_{x \sim P_{penalty}} [\max(0, \|\nabla_x D(x)\| - 1)] \}$$

$$(\|\nabla_x D(x)\| - 1)^2$$



ref:<https://arxiv.org/pdf/1704.00028.pdf>

# Least Squares GAN

$$\begin{aligned}
 \min_D V_{\text{LSGAN}}(D) &= \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [(D(\mathbf{x}) - b)^2] + \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [(D(G(\mathbf{z})) - a)^2] \\
 \min_G V_{\text{LSGAN}}(G) &= \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [(D(G(\mathbf{z})) - c)^2],
 \end{aligned}
 \tag{2}$$

ref:<https://arxiv.org/abs/1611.04076>



# Little Results



GAN result



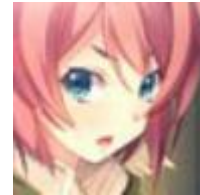
WGAN result

# Submission & Grading

# Data & format

- Anime Dataset

- training data: 33.4k (image, tags) pair
- **faces/**, tags.csv, sample\_testing\_text.txt



blue eyes  
red hair  
short hair

- training tags file format

- img\_id <comma> tag1 <colon> #\_post <tab> tag2 <colon> #\_post
- tags.csv is **not** used in HW3-1

```
1 0,touhou:17705 |chen:423 |moneti daifuku :60 |animal ears:12241 |catgirl:4903 |
2 1,touhou:17697 |onozuka komachi:224 |shikieiki yamaxanadu:217 |$
3 2,original:25774 |blonde hair:25457 |doll:1040 |dress:16585 |pink eyes:3896 |ta
4 3,amagi brilliant park:111 |musaigen no phantom world:39 |nichijou:142 |kawakan
```

tags.csv

- testing text file format

- testing\_text\_id <comma> testing\_text
- testing text only includes '**color hair**' and '**color eyes**', only alphabetic char involved.
- sample\_testing\_text.txt is **not** used in HW3-1

```
1 1,blue hair blue eyes
2 2,blue hair green eyes
3 3,blue hair red eyes
4 4,green hair blue eyes
```

sample  
testing\_text.txt

# Data & format

- Extra data

- training data: 36.7k (image, tags) pair
- **images/**, tags.csv



black eyes  
red hair

- training tags file format

- img\_id <comma> hair tag <space> eyes tag
- tags in extra data only includes 'color hair' and 'color eyes'
- tags.csv is **not** used in HW3-1

```
1 0,aqua hair aqua eyes
2 1,aqua hair aqua eyes
3 2,aqua hair aqua eyes
4 3,aqua hair aqua eyes
5 4,aqua hair aqua eyes
```

tags.csv

# Data Link

- [Anime Dataset](#)
- [Extra Data](#)

# HW3 Grading Policy:

- HW3-1 Code (image generation) : 5%
- HW3-2 Code (text-to-image generation): 5%
- Report : 15%
- HW3-3 (Bonus, style transfer): 2%
- 分工表 : 0.5%
- 上台分享 : 1%
- 上台分享前三名 : 1%

# HW3 Report Questions

- Model Description
  - Describe the models you use to, including the model architecture, objective function for G and D.
    - Image Generation (2%)
    - Text-to-image Generation (2%)
- Experiment settings and observation
  - Show generated images
    - Image Generation (1%)
    - Text-to-image Generation (1%)
- Compare your model with WGAN, WGAN-GP, LSGAN (choose 1) (**Image Generation Only**)
  - Model Description of the choosed model (1%)
  - Result of the model (1%)
  - Comparison Analysis (1%)
- Training tips for improvement (**Image generation Only**) (6%)

# Training Tips for improvement

- Pick **three** tips in the following website
  - <https://github.com/soumith/ganhacks>
  - Please implement these tips on image generation
- Total : 6%, 2% for each
  - Which tip & implement details (1%)
  - Result (image or loss...etc.) and Analysis (1%)
- Only the following tips are accepted
  - 1, 2, 3, 4, 5, 6, 9, 13, 14, 17



# HW3-1 Code Grading

- Reproduce Score : 2%
- Baseline Score : 2%
- TA Review : 1% (**mode collapse**)



# Output Format Requirement

- The generated images should be in Directory **samples/**
  - 請大家繳交時**就將產生的結果傳到samples/**, 助教利用script reproduce時也請同學將結果輸出到這個資料夾。為保證reproduce結果相同, 請同學將random的部份固定
  - 批改作業會在azure上, 請同學繳交前在機台上檢查
  - 已經在github裡的image -> samples/gan\_original.png
  - run\_gan.sh -> samples/gan.png
- Each generated image must be resized to **64 x 64**
- Generate 25 image into one png
  - sample code is in baseline.py
  - 為防止同學產生的圖片不一致, 請同學使用baseline.py裡的**save\_imgs()**

# Baseline Model

- [Anime Face Recognition](#)
- [Github for baseline.py](#)
- How to run:
  - pip install opencv-python
  - Download pre-trained model in [Here](#).
  - python baseline.py --input <input\_image>
- Generate 25 images in one png
  - if faces > 20: pass
- 下週二(5/8)會釋出最簡單架構

# Allow Packages

- Python 3.6
- **Tensorflow r.16 ONLY** (CUDA 9.0)
- PyTorch 0.3 / torchvision (**0.4 is not allowed**)
- Keras 2.0.7 (Tensorflow backend only)
- MXNet 1.1.0
- CNTK 2.4
- matplotlib
- skimage
- numpy, scipy
- Python Standard Library
- If you want to use other packages, please ask TAs for permission first!
- **new allowed package: pandas, tensorlayer, gensim, nltk**

# Submission on Github

- Only one branch **master** is needed
- Only **generator** and **inference** mode is needed
- Remember to put your **pre-trained models or download scripts** so that we can run your code successfully

# Submission

- Deadline: 2018/06/08 GMT+8 24:00
- **hw3/** should contain the following files:
  - **run\_gan.sh (hw3-1)**
  - **run\_cgan.sh (hw3-2)**
  - **extra\_run.sh (hw3-3, bonus)**
  - **samples/, samples/gan\_original.png**
  - **report.pdf**
  - **pre-trained model, python code...**
  - If some files are too big, upload to your cloud and download them when running your run.sh
- TAs will run your scripts in the following order to generate images
  - bash run\_gan.sh (hw3-1)
  - All scripts must output in **10 minutes**.
  - shell script裡面請寫**相對路徑**
- HW3-2, 3-3格式會在之後釋出

# Q&A

[ntu.mldsta@gmail.com](mailto:ntu.mldsta@gmail.com)