
MLDS HW3-2

TAs
ntu.mldsta@gmail.com

Update

- 5/15: 更新Baseline Model

Baseline Model (5/15 Update)

- Generator
 - noise_input = (100,);
 - text_input = (119,);
 - # num of (hair, eyes) pairs
 - text_emb = Dense(256,'relu')(text_input);
 - concatenate([noise_input, text_emb]);
 - Dense(4*4*512); Reshape((4, 4, 512));
 - Batchnorm(mom=0.9); Relu;
 - Conv2DTranspose(256, kernel=5);
 - Batchnorm(mom=0.9); Relu;
 - Conv2DTranspose(128, kernel=5);
 - Batchnorm(mom=0.9); Relu;
 - Conv2DTranspose(64, kernel=5);
 - Batchnorm(mom=0.9); Relu;
 - Conv2DTranspose(3, kernel=5);
 - Tanh;
- Training
 - Adam(lr = 0.0002, beta = 0.5)
- Discriminator
 - image_input = (64,64,3);
 - text_input = (119,);
 - text_emb = Dense(256,'relu')(text_input);
 - text_emb = Reshape((1,1,256))(text_emb);
 - tiled_emb = tile(text_emb, [1,4,4,1]);
 - Conv2D(64 ,kernel=5)(image_input); LeakyRelu;
 - Conv2D(128, kernel=5);
 - Batchnorm(mom=0.9); LeakyRelu;
 - Conv2D(256, kernel=5);
 - Batchnorm(mom=0.9); LeakyRelu;
 - Conv2D(512, kernel=5);
 - Batchnorm(mom=0.9);
 - image_feat = LeakyRelu;
 - concatenate([image_feat, tiled_emb]);
 - Conv2D(512, kernel=1, strides=(1,1));
 - 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

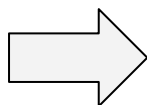
Schedule

- 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

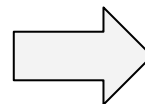
Task Descriptions

HW3-2: Text-to-Image Generation ^{1/2}

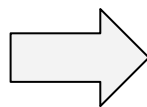
an all black bird



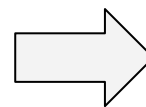
**Bird
Generative Model**



**this flower is
white and pink**

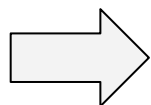


**Flower
Generative Model**

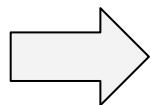


HW3-2: Text-to-Image Generation ^{2/2}

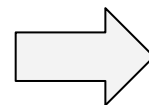
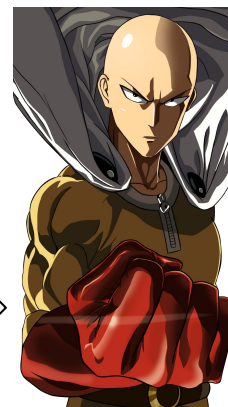
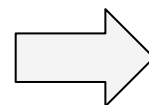
**a man with no hair
and one red strong fist**



**a girl with blue hair,
blue eyes and
twin ponytail**

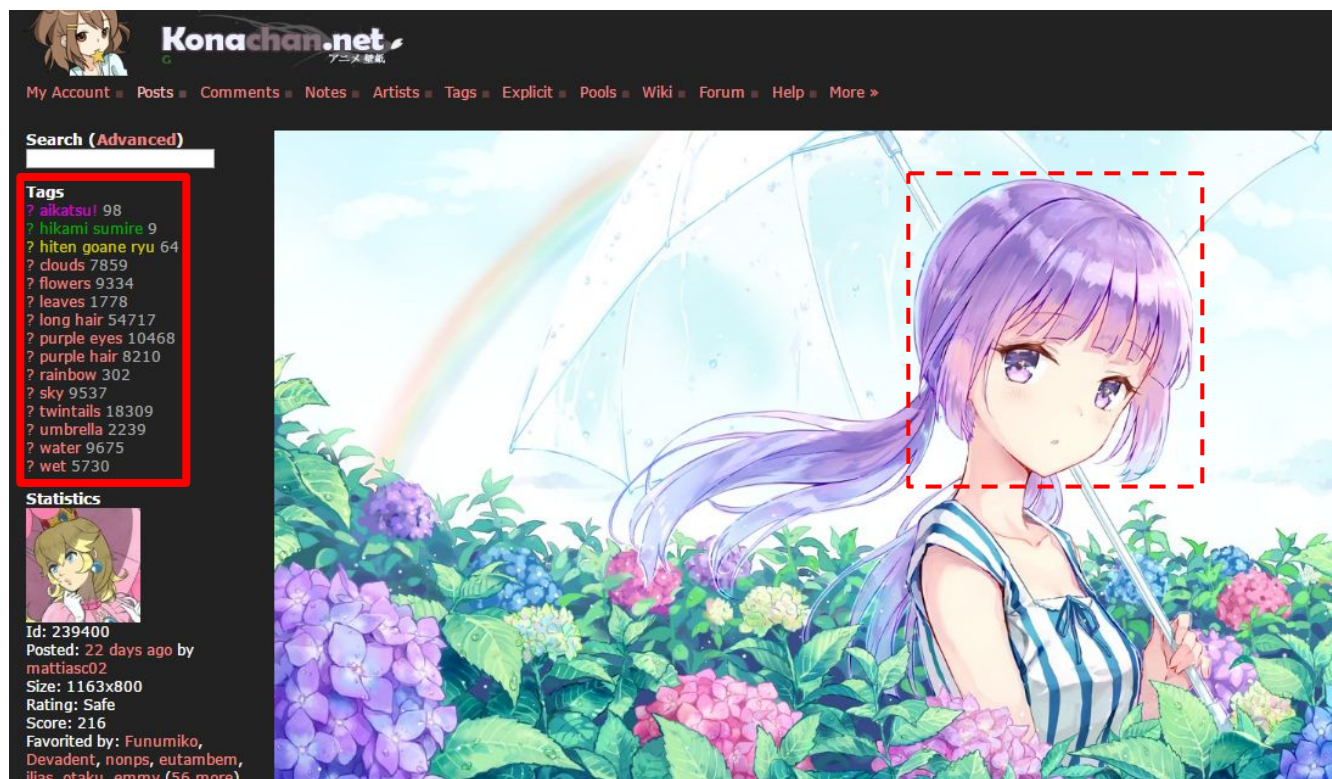


**Anime
Generative
Model**



Data Collections ^{1/2}

- Anime dataset




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 ^{2/2}

- 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



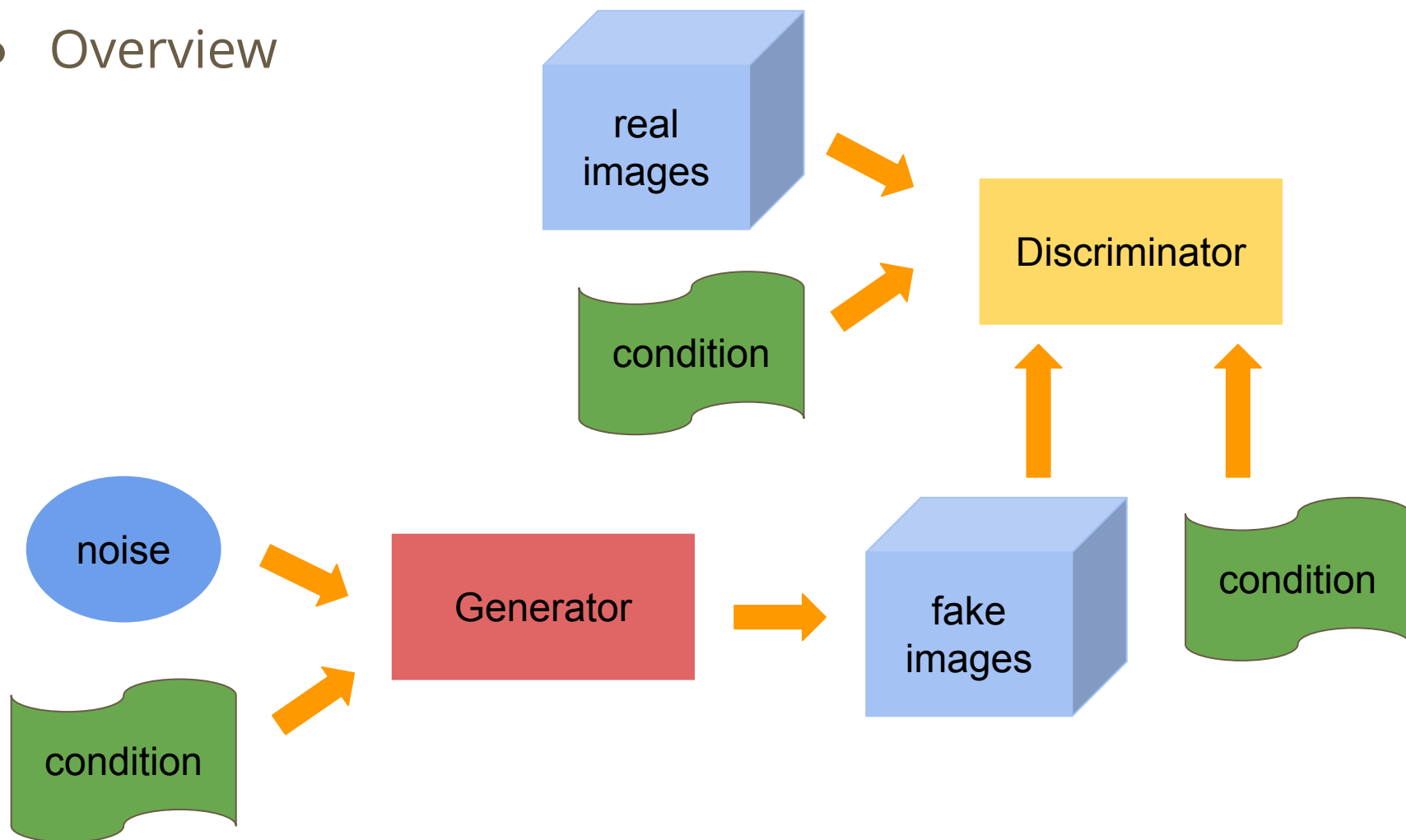
Noise Import/Export

Import Export

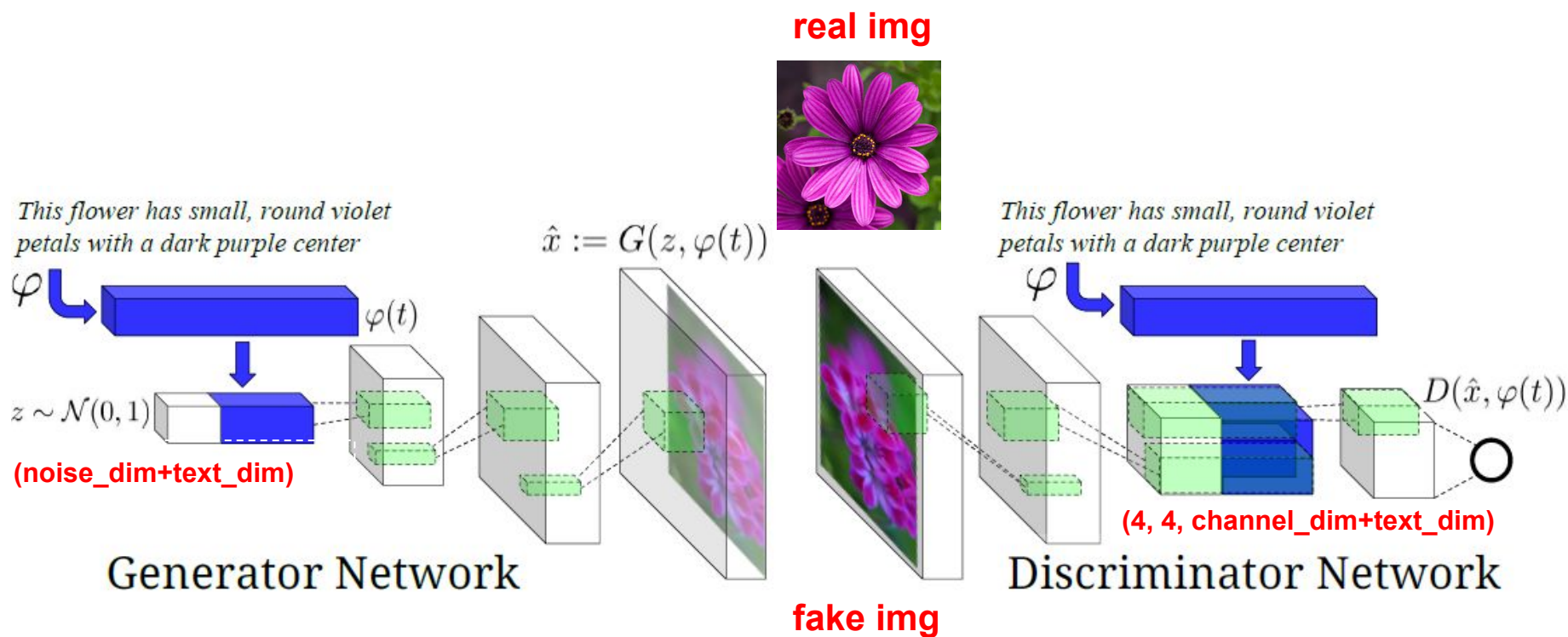
Model & Training Tips

Conditional GAN

- Overview

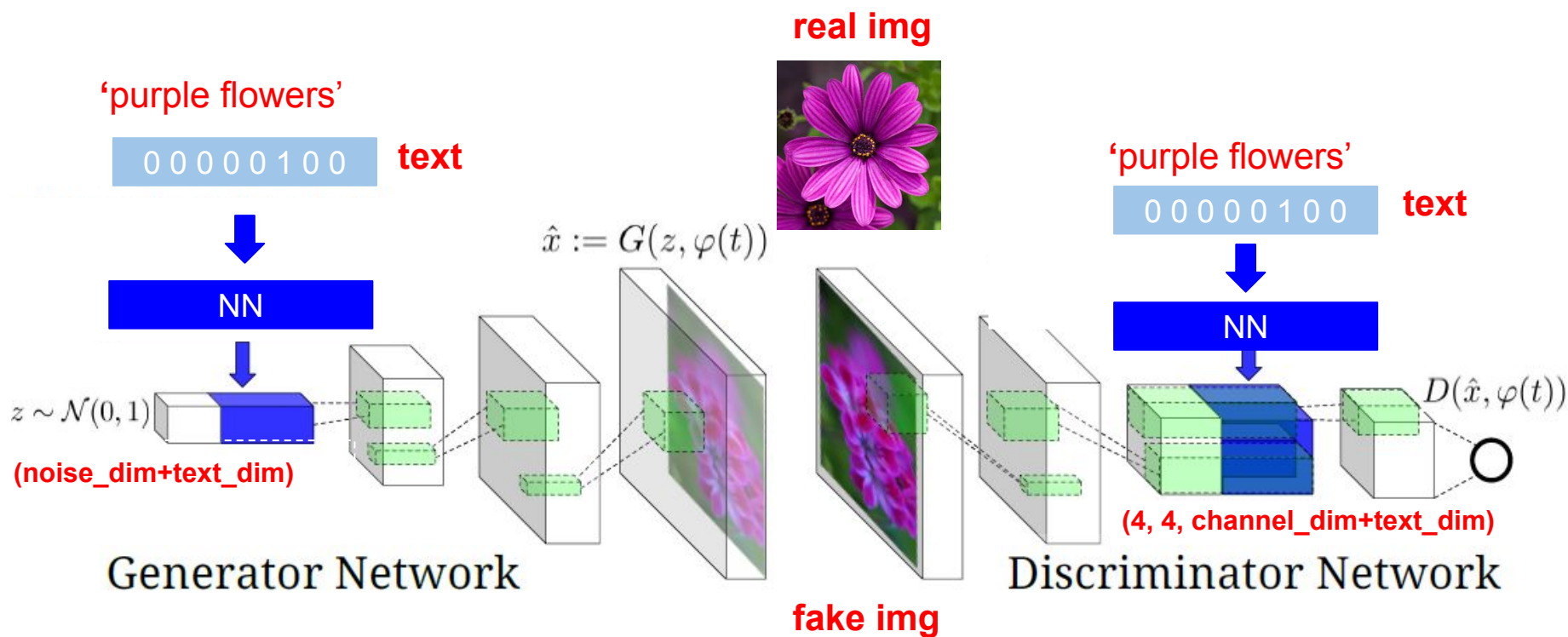


Conditional GAN for Text-to-Image Generation 1/2



ref: <https://arxiv.org/pdf/1605.05396.pdf>

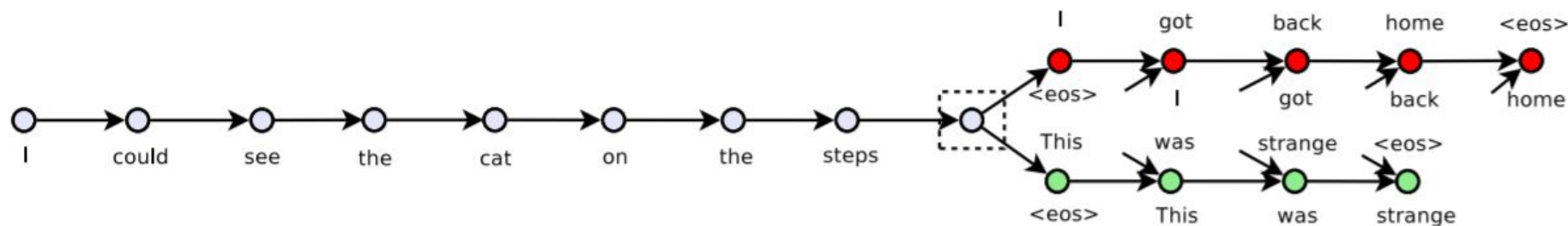
Conditional GAN for Text-to-Image Generation 2/2



ref: <https://arxiv.org/pdf/1605.05396.pdf>

Text Feature Process Tool - Skip-thought Vector

- Pretrained embedding



skip-thought source code:

https://github.com/tensorflow/models/tree/master/research/skip_thoughts#download-pretrained-models-optional

No matter which tool you use to process text input, please make sure you include that pre-trained model in your repository to let us run your code successfully.

Tip for Training

- Discriminator Output:
 - (real image, right text): 1
 - (fake image, right text): 0
 - (wrong image, right text): 0
- Different objective function
 - Wasserstein GAN (WGAN)
 - Improved WGAN (WGAN-GP)
 - Auxiliary Classifier GAN (ACGAN)
 - StackGAN

ACGAN _{1/2}

Discriminator should also be able to do a classification task.

$$L_{D,Q}^{ACGAN} = L_D^{GAN} + E[P(class = c|x)] + E[P(class = c|G(z))]$$

$$L_G^{ACGAN} = L_G^{GAN} + E[P(class = c|G(z))]$$

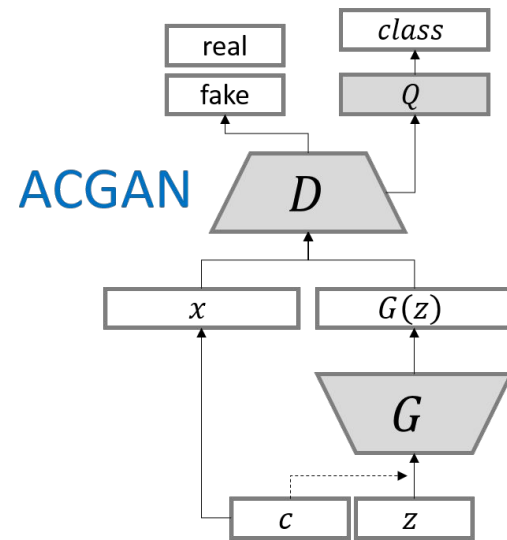
ACGAN _{2/2}

$$L_S = E[\log P(S = \text{real} \mid X_{\text{real}})] + \\ E[\log P(S = \text{fake} \mid X_{\text{fake}})]$$

$$L_C = E[\log P(C = c \mid X_{\text{real}})] + \\ E[\log P(C = c \mid X_{\text{fake}})]$$

D is trained to maximize $\mathbf{L}_s + \mathbf{L}_c$
while **G** is trained to maximize $\mathbf{L}_c - \mathbf{L}_s$

ref: <https://arxiv.org/pdf/1610.09585.pdf>



StackGAN

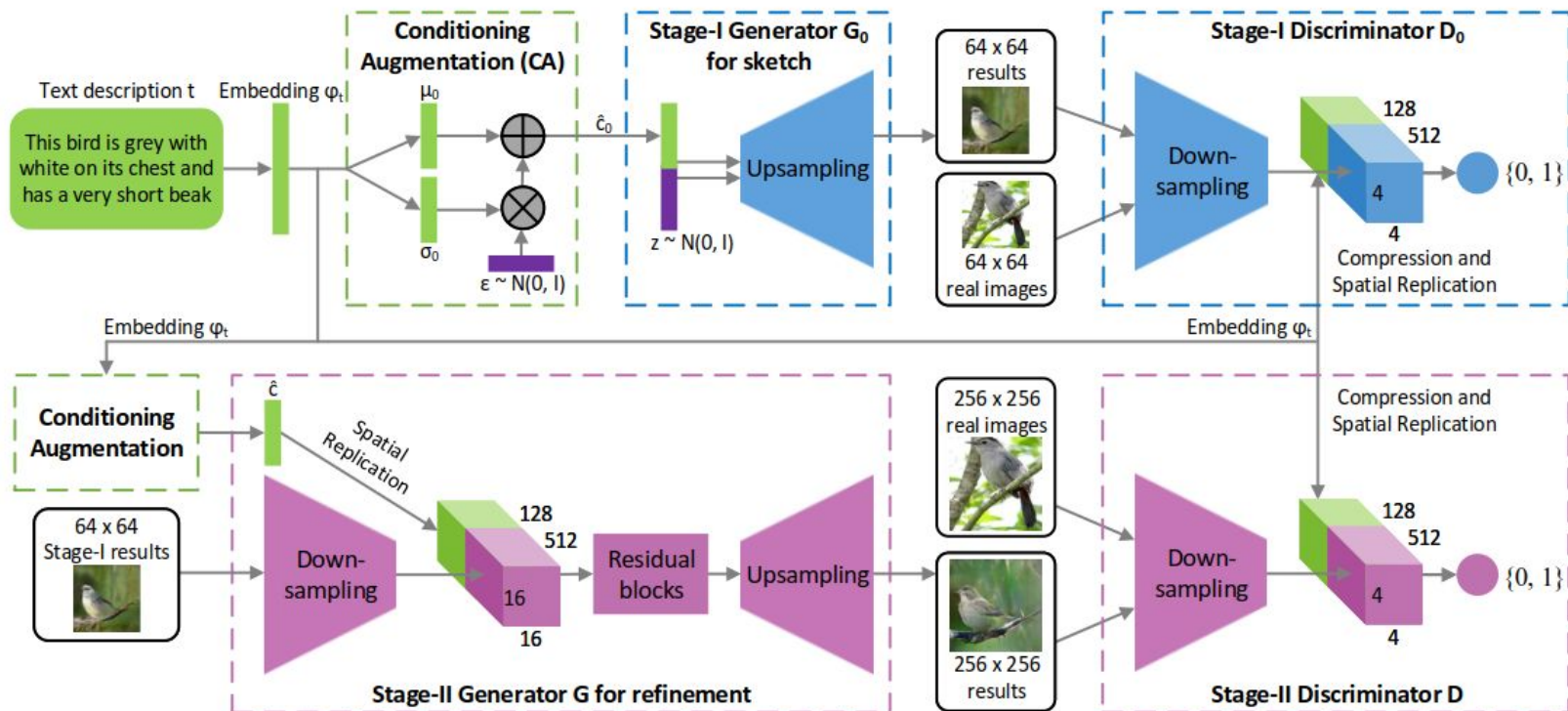


Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

ref:<https://arxiv.org/pdf/1612.03242.pdf>

Little Results

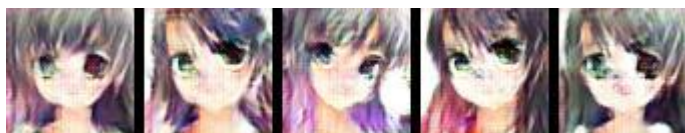
input text: black hair blue eyes



input text: pink hair green eyes



input text: green hair green eyes



input text: blue hair red eyes



GAN result

input text: black hair blue eyes



input text: pink hair green eyes



input text: green hair green eyes



input text: blue hair red eyes



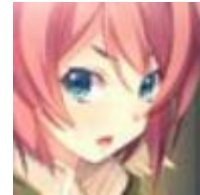
WGAN-GP result

Submission & Grading

Data & format

- Anime Dataset

- training data: 33.4k (image, tags) pair
- faces/, tags.csv, **testing_tags.txt**



blue eyes
red hair
short hair

- training tags file format

- img_id <comma> tag1 <colon> #_post <tab> tag2 <colon> #_post

```
1 0,touhou:17705 |chen:423 |moneti daifuku :60 |animal ears:12241 |catgirl:4903 |
2 1,touhou:17697 |onozuka komachi:224 |shikieiki yamaxonadu: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
5 4,original:25774 |blonde hair:25457 |doll:1040 |dress:16585 |pink eyes:3896 |ta
```

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.

```
1 1,blue hair blue eyes
2 2,blue hair green eyes
3 3,blue hair red eyes
4 4,green hair blue eyes
5
```


Testing Text Content

- **'color hair'**

- 'orange hair', 'white hair', 'aqua hair', 'gray hair', 'green hair', 'red hair', 'purple hair', 'pink hair', 'blue hair', 'black hair', 'brown hair', 'blonde hair'.

- **'color eyes'**

- 'gray eyes', 'black eyes', 'orange eyes', 'pink eyes', 'yellow eyes', 'aqua eyes', 'purple eyes', 'green eyes', 'brown eyes', 'red eyes', 'blue eyes'.

Data & format

- Extra data

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

- training tags file format

- img_id <comma> hair tag <space> eyes tag
- tags in extra data only includes 'color hair' and 'color eyes'



black eyes
red hair

```
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

All Testing Tags Content

- **'color hair'**

- 'orange hair', 'white hair', 'aqua hair', 'gray hair', 'green hair', 'red hair', 'purple hair', 'pink hair', 'blue hair', 'black hair', 'brown hair', 'blonde hair'.

- **'color eyes'**

- 'black eyes', 'orange eyes', 'pink eyes', 'yellow eyes', 'aqua eyes', 'purple eyes', 'green eyes', 'brown eyes', 'red eyes', 'blue eyes'.
- **no** 'gray eyes' in extra data

testing_tags.txt

1	1	,blue	hair	blue	eyes
2	2	,blue	hair	blue	eyes
3	3	,blue	hair	blue	eyes
4	4	,blue	hair	blue	eyes
5	5	,blue	hair	blue	eyes
6	6	,blue	hair	green	eyes
7	7	,blue	hair	green	eyes
8	8	,blue	hair	green	eyes
9	9	,blue	hair	green	eyes
10	10	,blue	hair	green	eyes
11	11	,blue	hair	red	eyes
12	12	,blue	hair	red	eyes
13	13	,blue	hair	red	eyes
14	14	,blue	hair	red	eyes
15	15	,blue	hair	red	eyes
16	16	,green	hair	blue	eyes
17	17	,green	hair	blue	eyes
18	18	,green	hair	blue	eyes
19	19	,green	hair	blue	eyes
20	20	,green	hair	blue	eyes
21	21	,green	hair	red	eyes
22	22	,green	hair	red	eyes
23	23	,green	hair	red	eyes
24	24	,green	hair	red	eyes
25	25	,green	hair	red	eyes

~~~~~

```
N ± master [1:testing_tags.txt ]
[1:testing_tags.txt ]
```

|      |       |    |    |     |
|------|-------|----|----|-----|
| unix | utf-8 | sh | 4% | 1:1 |
|------|-------|----|----|-----|

# 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%

# HW3-2 Code Grading

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

# 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/cgan\_original.png**
  - run\_cgan.sh -> **samples/cgan.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()**



# 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, samples/cgan\_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)
  - bash run\_cgan.sh **<testing\_tags.txt>** (hw3-2)
  - All scripts must output in **10 minutes**.
  - shell script裡面請寫**相對路徑**
- HW3-3格式會在之後釋出

# Q&A

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