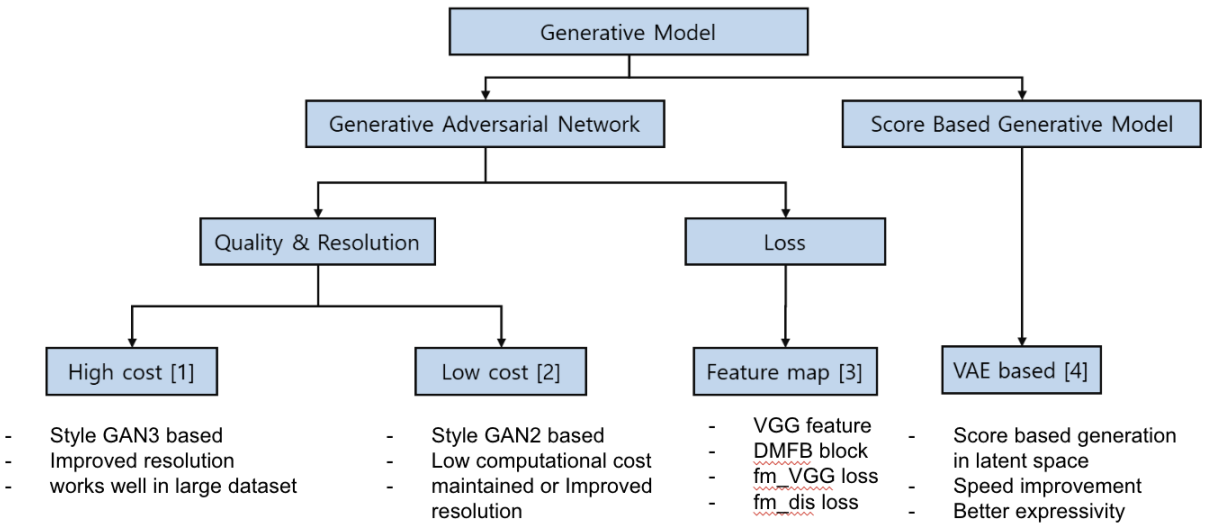
Deep Generative Models Report

20161190 이민재

1) Motivation and description for proposed idea

우리는 프로젝트를 수행하기 위해서 먼저 생성 모델 관련하여 각각의 다른 특징을 가진 최신 논문들을 각자 한편씩 읽어 리뷰 하였습니다. Style GAN3 기반으로 코스트가 높지만 고화질 이미지를 잘 생성해내고, 큰 데이터셋에서도 잘 작동하는 논문[1], Style GAN2 기반으로 낮은 코스트가 장점인 Anycost GAN 논문 [2], 새로운 VGG 를 이용하여 feature 를 추출하고, 새로운 block 과 loss 를 제안한 image fine grained inpainting 논문[3], VAE 를 기반으로한 Score based generative model 을 개선한 논문[4]을 읽었습니다. 위의 논문들을 읽어보니 특정한 한 모델을 기반으로 목적에 맞게 다른 모델들의 특징을 가져오거나, 새로운 특정을 추가로 정의하는 등의 방법들을 많이 보였습니다. Base model 을 정하기 위해 Image inpainting 을 수행하는 여러 모델들을 돌려보았습니다. 그 결과 우리는 하나의 문제점을 발견하였습니다. 원본 이미지는 웃고있는 모습이지만, inpainting 을 할 경우 무표정과 가까운 이미지가 생성되는 등 표정,감정의 특성이 잘 반영되지 않는 현상이 발생하였습니다. 저희는 평가 지표들도 중요하지만 이러한 특성 또한 중요하다고 생각하였습니다. 그래서 이를 보안하기 위해 저희는 landmark 와 emotion 을 새롭게 loss 에 추가해보자는 생각을 하였습니다. 또한 추후 실험을 진행하면서, 이들을 각각 모듈화 하여 필요성에 따라 추가하거나 제거하고 이미지를 생성할 수 있는 기능도 추가하기로 하였습니다. 테스트 데이터셋을 받았을때, 일반적인 데이터셋 보다 상대적으로 한국인의 데이터가 많다고 생각했습니다. 그래서 기본적으로 대표적인 얼굴 데이터셋인 FFHQ 나 CelebA 를 사용하되, 동양인 데이터셋을 따로 추가하여 학습시키기로 하였습니다.



<그림 1. 논문들의 특징>

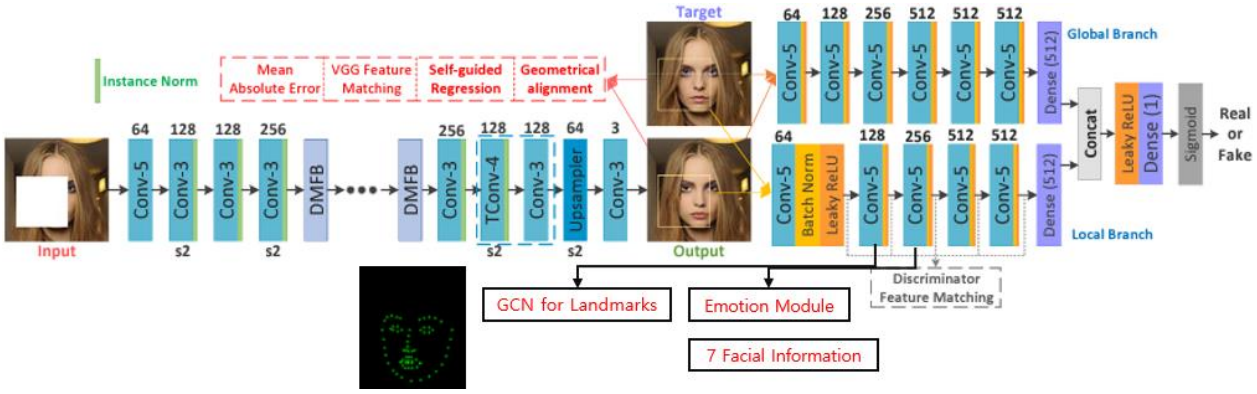
2) Detailed method. Description

위에서 설명하였듯이, 저희는 테스트 데이터셋에 한국인의 비율이 상대적으로 높아 기본적인 데이터셋에 동양인의 사진을 추가하여 학습시키기로 하였습니다. 하지만 동양인만 모아둔 얼굴 데이터셋을 따로 찾기가 힘들었고, 크롤링을 또는 수작업을 통해서 해야한다는것을 깨달았습니다. 결국 수작업으로 캡쳐를 하거나, 대표적 데이터셋에서 동양인만 따로 추출하여 추가적인 데이터셋을 만들었습니다.

여러 모델들을 테스트한 결과, 상대적으로 가볍고 지표 점수가 잘 나오는 Fine grain inpainting 모델과 Lafin 모델을 이용하여 모델을 설계하기로 하였습니다. 모델의 성능과 속도는 모델의 평가에 중요한 요소입니다. 하지만 이는 대부분 상황에서 비례하지 않고 모델에 따라 정해진 경우가 많습니다. 우선 모델의 Generatorf 를 최대한 가볍게 하기 위해 parameter 의 수는 고정하였습니다. 또한, Motivation 에서 설명하였듯이, 저희가 제안한 landmark 모델과 emotion 모델을 각각 모듈화 하여 유저가 적재적소에 추가/제거를 할수 있어 상황에 맞게 성능과 속도를 선택할 수 있게 하였습니다. Landmark 와 emotion 을 고려하여 수식으로 표현하면 다음과 같습니다.

$$G_\theta(x, l, e) = X \;\rightarrow\; G_\theta(x|l', e') = X$$

G_θ는 generator 의 learning parameter 입니다. x 는 삽입하는 이미지, l 은 landmark 정보, e 는 emotion 정보, X 는 결과 이미지이고, l'은 discriminator 를 통한 landmark 정보의 조건부 확률, e'는 discriminator 를 통한 emotion 정보의 조건부 확률입니다. 또한 우리는 landmark 를 조금더 효율적으로 적용하기 위해 GCN 과 CNN 을 각각 적용하여 테스트 하였으며, 실험결과 landmark 를 사용할때 GCN 을 적용하였습니다.



<그림 2. 제안한 모델구조>

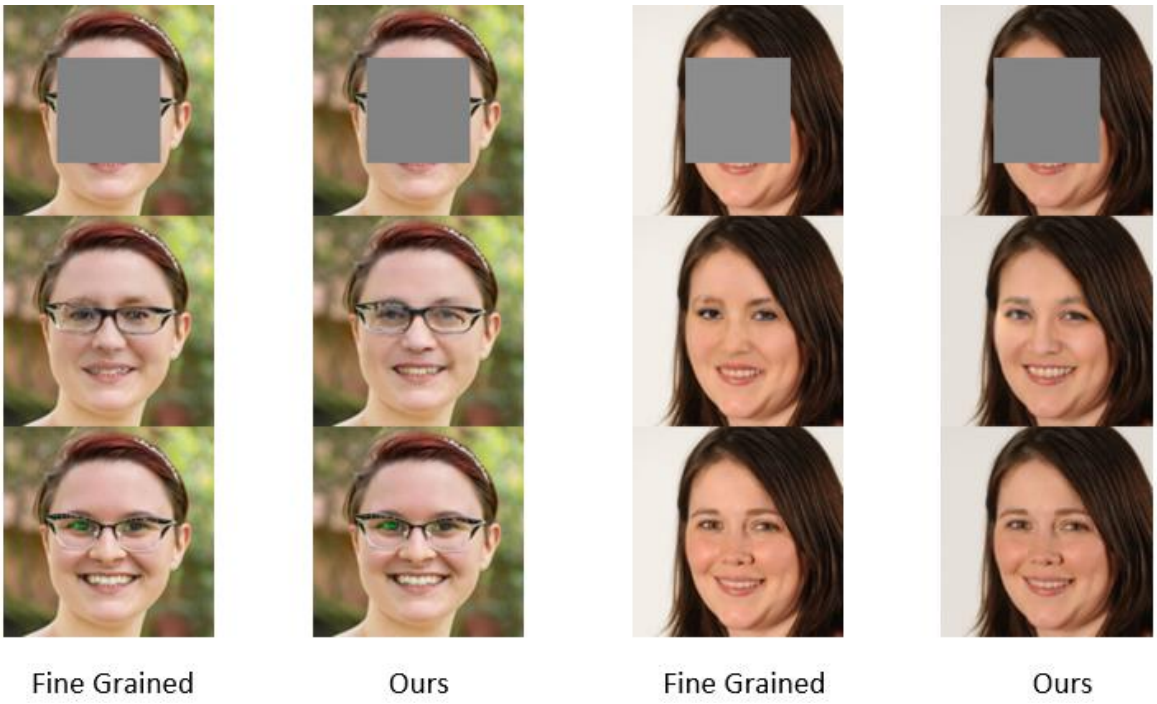
3) Presentation and analysis on the result

저희는 Lafin 모델과, 우리의 모델의 실행 시간과 parameter 개수를 비교해 보았습니다(그림 3). 분석 결과 Lafin 모델의 실행시간은 저희 모델보다 약 5 배정도 시간이 더 걸렸으며, parameter 의 수는 6.3 배정도 많이 사용하였습니다. 실제 모델을 정하기 전, 다양한 inpainting 모델들을 테스트하여 실행 시간의 중요성을 느꼈기 때문에, 의미 있는 결과라 생각합니다. 또한 저희의 main contribution 인 emotion

loss 를 추가하여 학습한 것과 Fine grained inpainting 모델과 비교해 보았습니다(그림 4). 분석 결과 Fine grained 모델은 생성된 이미지를 보면 미소 짓고 있는 input 의 emotioin 특징은 잘 잡아내지 못한 결과를 보여주고 있습니다. 반면 저희 모델은 emotion loss 를 추가하여 학습하였기 때문에, 상대적으로 emotion 특징을 잘 잡아내는 결과를 보여줍니다. Inpainting 을 할 때 사람의 얼굴 부분의 비율, 크기 등을 잘 잡아내는 것도 중요하지만, 사람의 emotion 을 잡아 내는 것 또한 중요하다고 생각합니다. 이는 단순한 평가지표의 점수가 높게 나오는 것 이상의 의미를 가진다고 생각합니다.

model	Inference time	Total parameter
lafin	0.576719	57582768
ours	0.113357	9037443
improvement	508.76%	637.16%

<그림 3. 우리의 모델과 Lafin 모델의 실행시간, parameter 수 비교>



<그림 4. 우리의 모델과 Fine Grained 모델의 결과비교>

4) The evaluation scores (PSNR, SSIM, LPIPS)

모델을 평가할 때 사용한 평가지표는 총 3 가지(PSNR, SSIM, LPIPS) 입니다. PSNR 은 일반적으로 이미지나 영상을 압축했을 때 화질이 얼마나 손실되었는지 평가하는 목적으로 사용합니다. 데시벨(db) 단위를 사용하며, PSNR 수치가 높을수록 원본 영상에 비해 손실이 적다는 의미입니다. SSIM 은 PSNR 과 다르게 수치적인 에러가 아닌 인간의 시각적 화질 차이를 평가하기 위해 고안된 방법입니다. 휘도, 대비, 구조 3 가지 기준으로 평가합니다. SSIM 과 마찬가지로, 낮을수록 성능이 좋다는 의미입니다. LPIPS 는 위의 위의 두 지표로는 시각적 질을 평가하기에 부족하다 판단해 이미지의 시각적인 특징이 유사한지 아닌지 집중하는 지표 입니다. 그렇기 때문에 인간이 판단하는 기준에 가까운 지표입니다. LPIPS 는 pretrained Alexnet 을 활용해 image features 를 추출하고 두 피쳐간 거리를 계산합니다. 처음의 두 지표와는 달리, 낮을수록 성능이 우수하다는 의미입니다.

저희가 모델에서 고려한 요소는 총 3 가지(Feature, Emotion, Landmark) 입니다. 이들은 모두 성능 향상에 도움을 줄 것이라고 생각하였고, 각각 얼마나 평가 지표 상승에 기여를 하는지 보기 위해 모든 경우의 수를 고려한 8 가지 테스트 해보았습니다. 그 결과 생각한것과는 다르게 vgg feature 과 landmark feature 을 동시에 사용하는 경우는 점수가 낮게 나오는 것을 확인하였습니다. 표를 보면, vgg feature 를 사용, emotion feature 를 사용, landmark feature 를 사용하지 않은 7 번째 모델이 가장 높은 점수를 얻었습니다.

Test method	Feature	Emotion	Landmark graph	PSNR(high)	SSIM(high)	LPIPS(low)
Colab	X	X	X	36.3681	0.8905	0.0795
Colab	X	X	O	36.2563	0.8954	0.077
Colab	X	O	X	36.2033	0.8932	0.0778
Colab	O	X	X	37.006	0.9107	0.0572
Colab	X	O	O	36.1976	0.8929	0.0786
Colab	O	X	O	36.9553	0.9082	0.0601
Colab	O	O	X	36.9784	0.9104	0.0565
Colab	O	O	O	36.8857	0.9094	0.059

5) Description of the source code

Train 과 test 는 각각 python train.py python test.py 명령어를 입력하여 진행할수 있습니다. 여러가지 폴더들이 있는데, config 파일과 arg 값을 변경하며 다양한 실험을 하였습니다. Models 폴더를 보면, 총 8 가지 구조와 모델이 있습니다. 이는 각각 3 가지의 요소인 feature, emotion, landmark 를 각각 사용/사용하지 않았을 경우의 모델들입니다.

아래의 코드는 train2.py 소스 코드입니다.

```
import argparse
import os
from tensorboardX import SummaryWriter
from utils import get_config, prepare_sub_folder, _write_images, write_html
from data import create_dataset, create_dataloader
import math
from models.inpainting_model8 import InpaintingModel
import torch

parser = argparse.ArgumentParser()
# parser.add_argument('--config', type=str, default='configs/paris-celeba-hq-regular_list.yaml', help='path to the config file.')

parser.add_argument('--config', type=str, default='/home2/projects/DGM3/DMFN_ver5/configs/ffhq_list.yaml', help='path to the config file.')
parser.add_argument('--output_path', type=str, default='/home2/projects/DGM3/DMFN_ver5/', help='output path of the tensorboard file')
args = parser.parse_args()

os.environ['CUDA_VISIBLE_DEVICES'] = '3'
config = get_config(args.config)
torch.backends.cudnn.benchmark = True
```

먼저 모듈들을 불러오고, parser.add_argument 로 받아들일 인수들을 추가해 나갑니다.

```
model_name=os.path.splitext(os.path.basename(args.config))[0].split('_')[0]
train_writer = SummaryWriter(os.path.join(args.output_path + '/logs', model_name))

output_dir = os.path.join(args.output_path + '/outputs', model_name,'featureandemotionandland8')
checkpoint_dir, image_dir = prepare_sub_folder(output_dir)
config['checkpoint_dir'] = checkpoint_dir
```

실시간으로 학습과정을 확인하기 위해 tensorboard 를 사용합니다.

```
for phase, dataset_opt in config['datasets'].items():
    if phase == 'train':
        train_set = create_dataset(dataset_opt)
        train_size = int(math.ceil(len(train_set) / dataset_opt['batch_size']))
        print('Number of training images: {:,d}, iters: {:,d}'.format(len(train_set), train_size))
        total_iters = int(dataset_opt['n_iter'])
        total_epochs = int(math.ceil(total_iters / train_size))
        print('Total epochs needed: {:d} for iters {:,d}'.format(total_epochs, total_iters))
        train_loader = create_dataloader(train_set, dataset_opt)
    elif phase == 'val':
        val_set = create_dataset(dataset_opt)
        val_loader = create_dataloader(val_set, dataset_opt)
        print('Number of val images in [{:s}]: {:d}'.format(dataset_opt['name'], len(val_set)))
    elif phase == 'test':
        pass
    else:
        raise NotImplementedError('Unsupported phase: {:s}'.format(phase))
```

train 과 val 데이터로더를 생성합니다.

```
model = InpaintingModel(config)
start_epoch = 0
current_step = 0
```

모델을 생성합니다.

```
for epoch in range(start_epoch, total_epochs):
    for _, train_data in enumerate(train_loader):
        current_step += 1
        if current_step > total_iters:
            break

        model.update_learning_rate()
```

```
            model.feed_data(train_data)
            model.optimize_parameters()


            if current_step % config['log_iter'] == 0:
                logs = model.get_current_log()
                message = '[epoch:{:3d}, iter:{:8,d}, lr:{:.3e}] '.format(epoch, current_step,
                                                                          model.get_current_learning_rate())
                for k, v in logs.items():
                    message += '{:s}: {:.4f} '.format(k, v)

                    train_writer.add_scalar(k, v, current_step)
                print(message)
                log_path=os.path.join(output_dir, 'training.txt')
                f = open(log_path, 'a')
                f.write(message+'\n')
                f.close()

            if current_step % config['val_iter'] == 0:
                v_input, v_output, v_target = [], [], []
                visual_images = []
                for index, val_data in enumerate(val_loader):
                    if index < config['display_num']:
                        model.feed_data(val_data)
                        model.test()
                        visuals = model.get_current_visuals()
                        v_input.append(visuals['input'])
                        v_output.append(visuals['output'])
                        v_target.append(visuals['target'])
                    else:
                        break

                visual_images.extend(v_input)
                visual_images.extend(v_output)
                visual_images.extend(v_target)
                _write_images(visual_images, config['display_num'], '%s/val_current.jpg' % image_dir)


            if current_step % config['save_image_iter'] == 0:
                v_input, v_output, v_target = [], [], []
                visual_images = []
                for index, val_data in enumerate(val_loader):
                    if index < config['display_num']:
                        model.feed_data(val_data)
                        model.test()
                        visuals = model.get_current_visuals()
                        v_input.append(visuals['input'])
                        v_output.append(visuals['output'])
                        v_target.append(visuals['target'])
                    else:
                        break

                visual_images.extend(v_input)
                visual_images.extend(v_output)
                visual_images.extend(v_target)
                _write_images(visual_images, config['display_num'], '%s/val_%08d.jpg' % (image_dir, current_step))

                write_html(output_dir + '/index.html', current_step, config['save_image_iter'], 'images')


            if current_step % config['save_model_iter'] == 0:
                print('Saving models.')
                model.save(current_step)

print('Saving the final model.')
model.save('latest')
print('End of training.')
```

모델을 training 시킨후 모델을 저장합니다. (학습률 업데이트, 훈련, 검증, 이미지와 html 파일 저장, 모델 저장)

아래의 코드는 test2.py 소스 코드입니다.

```
import argparse
import os
from utils import get_config, _write_images
import torch
from data import create_dataset, create_dataloader
from models.networks2 import define_G
from data.util import tensor2img
import skimage.io as sio
import numpy as np

def createFolder(directory):
        try:
                if not os.path.exists(directory):
                        os.makedirs(directory)
        except OSError:
                print ('Error: Creating directory. ' +    directory)

os.environ["CUDA_VISIBLE_DEVICES"] = '1'
parser = argparse.ArgumentParser()
parser.add_argument('--config', type=str, default='configs/ffhq_list.yaml', help="net configuration")
parser.add_argument('--output_folder', type=str, default='outputs/ffhq/featureandemotionandland2/test', help="output image path")
parser.add_argument('--checkpoint', type=str, default='outputs/ffhq/featureandemotionandland2/checkpoints/100000_G.pth',
                                help="checkpoint of generator")
opts = parser.parse_args()

if not os.path.exists(opts.output_folder):
        os.makedirs(opts.output_folder)

config = get_config(opts.config)
device = torch.device('cuda')

model = define_G(config).to(device)
model.load_state_dict(torch.load(opts.checkpoint), strict=True)
model.eval()

print('Loading the checkpoint for G [{:s}] ...'.format(opts.checkpoint))
```

마찬가지로, 모듈들을 불러온후, parser.add_argument 를 사용하여 받아들일 인수들을 추가해 나갑니다. 그후 실험 환경을 불러오고, 모델과 데이터로더를 세팅합니다.

```
with torch.no_grad():
        dataset_opt = config['datasets']['test']
        test_set = create_dataset(dataset_opt)
        test_loader = create_dataloader(test_set, dataset_opt)
        print('Number of test images in [{:s}]: {:d}'.format(dataset_opt['name'], len(test_set)))


        for index, test_data in enumerate(test_loader):
                v_input, v_output, v_target = [], [], []
                visual_images = []
                var_input, var_mask, var_target, img_paths = test_data['input'], test_data['mask'], test_data['target'], \
                                                                test_data['paths']
                var_input = var_input.to(device)
                var_mask = var_mask.to(device)
                var_target = var_target.to(device)
                var_output = var_mask.detach() * model(torch.cat([var_input, var_mask], dim=1)) + (
                                1 - var_mask.detach()) * var_input.detach()
                v_input.append(var_input.detach()[0].float().cpu())
                v_output.append(var_output.detach()[0].float().cpu())
                v_target.append(var_target.detach()[0].float().cpu())
                visual_images.extend(v_input)
                visual_images.extend(v_output)
```

```
            visual_images.extend(v_target)
            _write_images(visual_images, 1, '%s/%s' % (opts.output_folder, img_paths[0].split('/')[-1]))
            saved_mask = (var_mask.detach()[0].float().cpu().numpy().squeeze() * 255).round().astype(np.uint8)
            saved_input = (var_mask.detach()[0].float().cpu() + ((v_target[0] + 1) / 2)).numpy().squeeze().transpose(1, 2, 0).clip(0, 1)
            saved_output = tensor2img(v_output)
            saved_target = tensor2img(v_target)
            createFolder(os.path.join(opts.output_folder, 'mask'))
            createFolder(os.path.join(opts.output_folder, 'input'))
            createFolder(os.path.join(opts.output_folder, 'output'))
            createFolder(os.path.join(opts.output_folder, 'target'))
            sio.imsave(os.path.join(opts.output_folder, 'mask', img_paths[0].split('/')[-1].split('.')[0] + '.png'), saved_mask)
            sio.imsave(os.path.join(opts.output_folder, 'input', img_paths[0].split('/')[-1]), saved_input)
            sio.imsave(os.path.join(opts.output_folder, 'output', img_paths[0].split('/')[-1]), saved_output[0])
            sio.imsave(os.path.join(opts.output_folder, 'target', img_paths[0].split('/')[-1]), saved_target[0])

print('End of testing.')
```

테스트 데이터를 불러와 모델을 테스트 합니다. 그후 생성된 이미지를 저장합니다.

6) Contribution table and Reference

Team member name	Role and responsibility	Contribution score
이민재	Data collection, Loss function review, 1 paper review, PPT format and presentation, Team Leader	25%
오영훈	6 papers review and share, Related work, PPT review paper part writes, Brainstorming,	25%
유지태	4 papers review and share, Introduction work, PPT remain part writes Brainstorming	25%
추영준	4 papers review and share, Experiment part work, Idea proposes, PPT experiment part writes, Implementation	25%

[1] Sauer, A., Schwarz, K., & Geiger, A. (2022). Stylegan-xl: Scaling stylegan to large diverse datasets. arXiv preprint arXiv:2202.00273.

[2] Lin, J., Zhang, R., Ganz, F., Han, S., & Zhu, J. Y. (2021). Anycost gans for interactive image synthesis and editing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14986-14996).

[3] Hui, Z., Li, J., Wang, X., & Gao, X. (2020). Image fine-grained inpainting. arXiv preprint arXiv:2002.02609.

[4] Vahdat, A., Kreis, K., & Kautz, J. (2021). Score-based generative modeling in latent space. Advances in Neural Information Processing Systems, 34.

[5] Yang, Y., Guo, X., Ma, J., Ma, L., & Ling, H. (2019). Lafin: Generative landmark guided face inpainting. arXiv preprint arXiv:1911.11394.
[6] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
[7] Wen, Z., Lin, W., Wang, T., & Xu, G. (2021). Distract Your Attention: Multi-head Cross Attention Network for Facial Expression Recognition. arXiv preprint arXiv:2109.07270.

(코드와 데이터셋은 추영준님이 제출할 예정입니다)