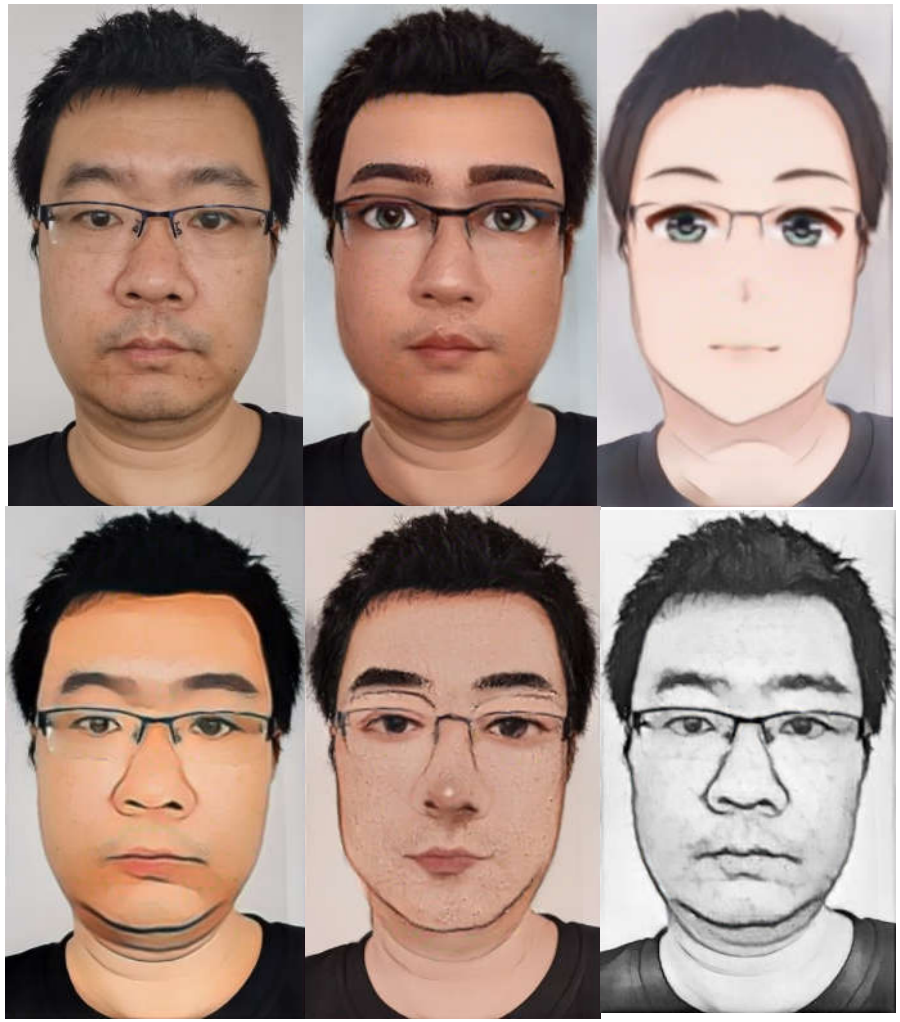


Python编程与人工智能实践

应用篇：人物卡通化

DCT-Net:
Domain-Calibrated Translation

于泓
鲁东大学
信息与电气工程学院

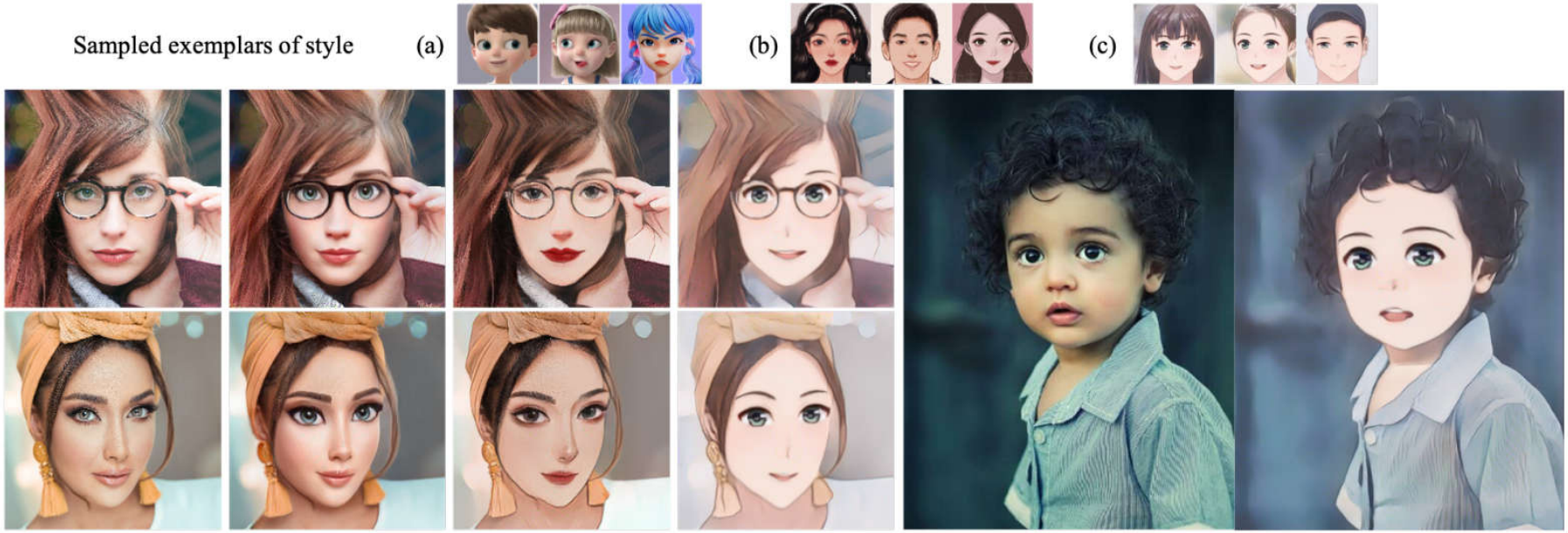
2022.11.5

人物卡通化

DCT-Net: Domain-Calibrated Translation for Portrait Stylization

YIFANG MEN*, DAMO Academy, Alibaba Group, China
YUAN YAO, DAMO Academy, Alibaba Group, China
MIAOMIAO CUI, DAMO Academy, Alibaba Group, China
ZHOUHUI LIAN*, Wangxuan Institute of Computer Technology, Peking University, China
XUANSONG XIE, DAMO Academy, Alibaba Group, China

项目网址:
https://github.com/menyifang/DCT-Net.

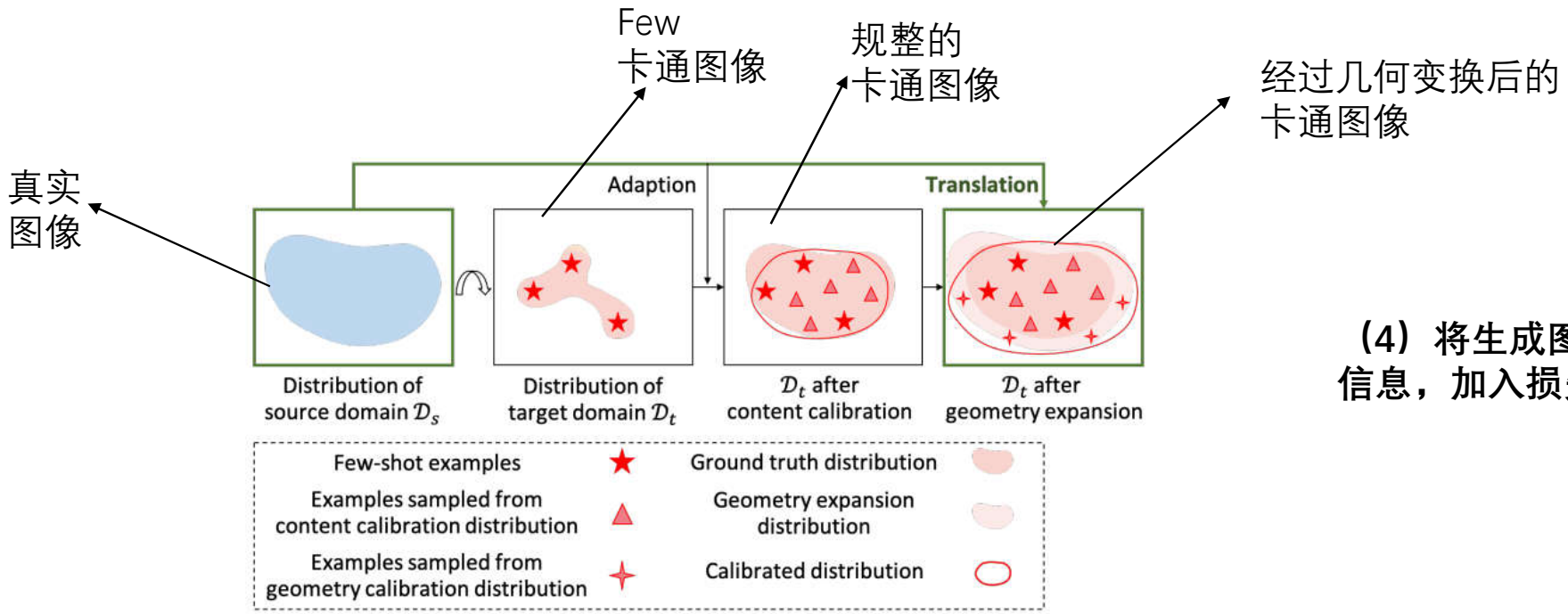


Input | (a) 3D cartoon | (b) Hand-drawn | (c) Anime | Full-body image (input) | Anime (output)

项目特点:

（1）few-shot learning 只需要几百张图片就可以实现风格的转换

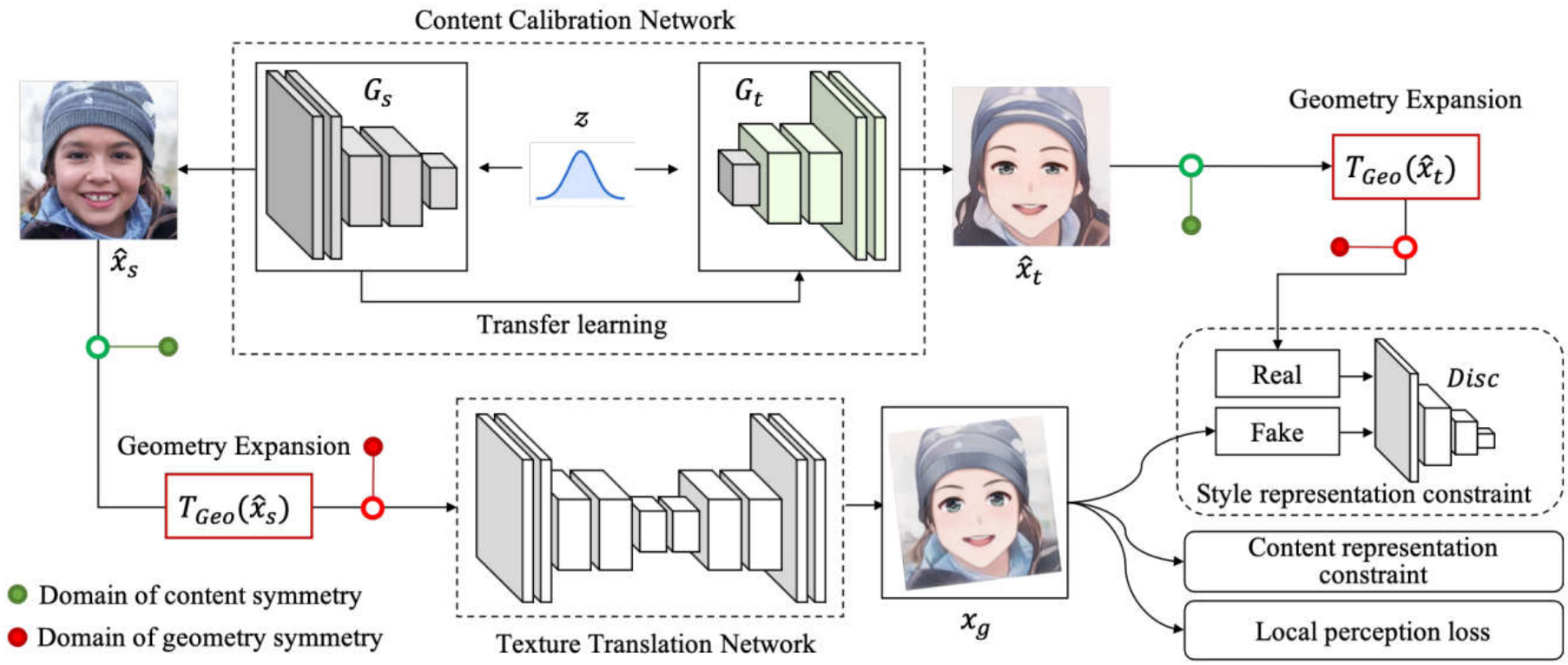
（2）提出了 Domain-Calibrated 的概念



（4）将生成图像的人脸信息，加入损失函数

（3）提出了几何扩展模型（Geometry expansion module ）

模型整体结构


Content Calibration Network
G_s
z
G_t
Transfer learning
$\hat{x}_s$
$\hat{x}_t$
Geometry Expansion
$T_{Geo}(\hat{x}_t)$
Real
Fake
Disc
Style representation constraint
Geometry Expansion
$T_{Geo}(\hat{x}_s)$
Texture Translation Network
x_g
Content representation constraint
Local perception loss
Domain of content symmetry
Domain of geometry symmetry

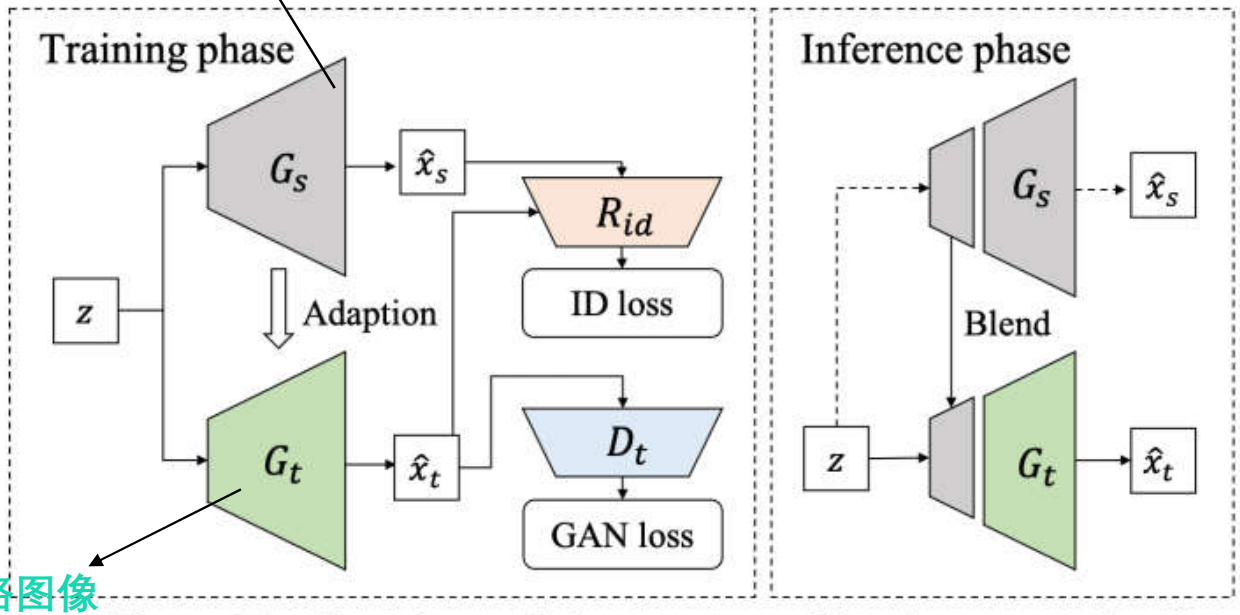
三个主要模块：

（1）内容校准网络

content calibration network (CCN)

一个训练好的 StyleGAN2

迁移学习



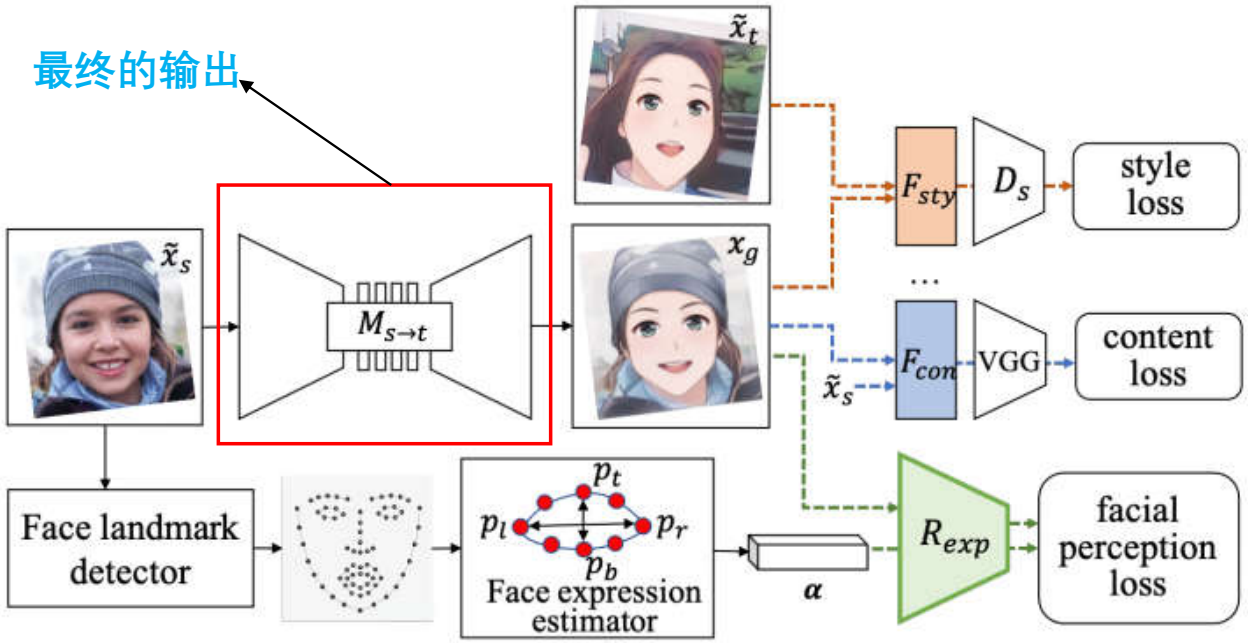
生成风格图像

Fig. 4. The flowchart of the content calibration network.

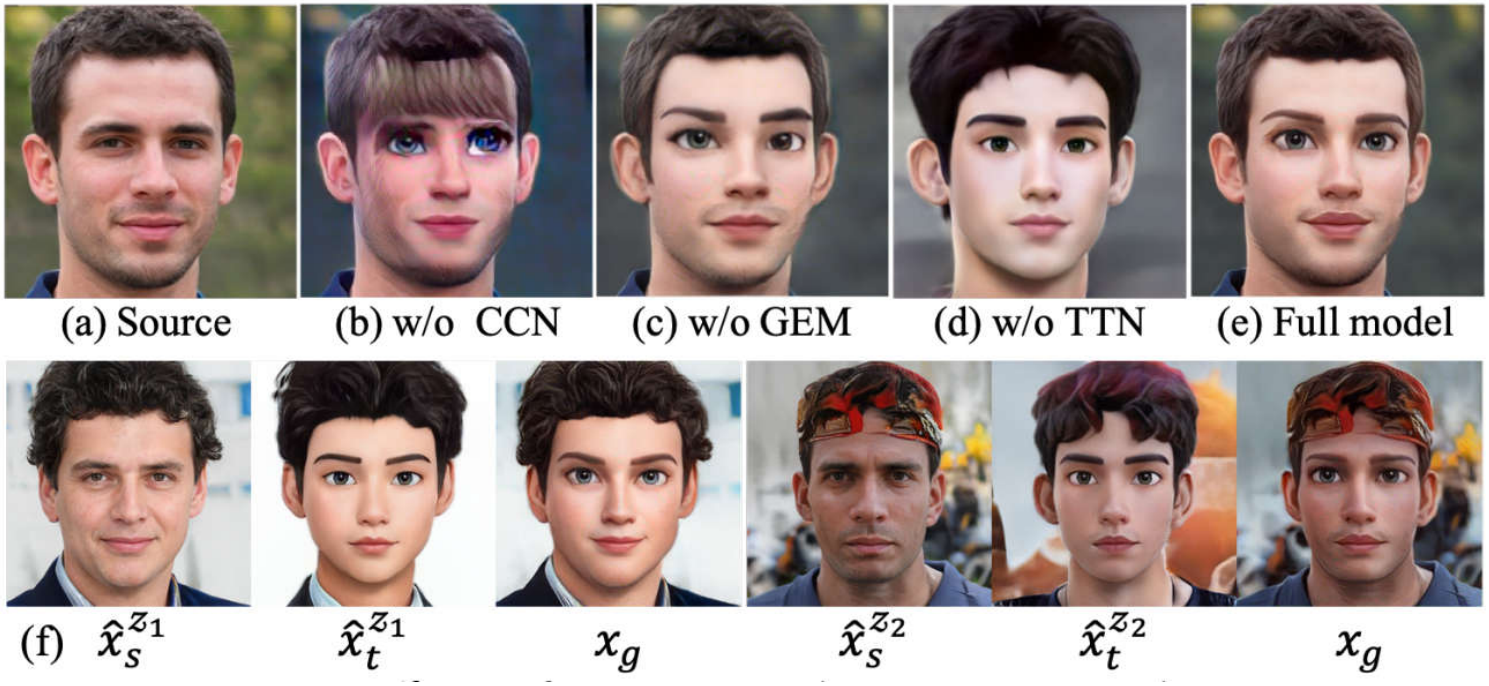
（2）几何扩展模型（Geometry expansion module ） 对图像进行随机变换，增加样本的丰富性

$\hat{x}_t \longrightarrow \overline{\tilde{x}_t}$

（3）纹理变换网络



一些对比结果


(a) Source
(b) w/o CCN
(c) w/o GEM
(d) w/o TTN
(e) Full model
(f) $\hat{x}_s^{z_1}$
$\hat{x}_t^{z_1}$
x_g
$\hat{x}_s^{z_2}$
$\hat{x}_t^{z_2}$
x_g

代码实现：

https://modelscope.cn/home

（1） 利用modelscope 来实现

安装 modelscope

```
pip install "modelscope[audio,cv,nlp,multi-modal,science]" -f https://modelscope.oss-cn-beijing.aliyuncs.com/releases/repo.html
```

```
from modelscope.pipelines import pipeline
from modelscope.utils.constant import Tasks

p = pipeline('image-portrait-stylization', 'damo/cv_unet_person-image-cartoon_compound-models')
```

（2）1. 手动下载源代码 https://github.com/menyifang/DCT-Net

2. 手动下载模型

- cartoon_anime
- cartoon-3d
- cartoon-artstyle
- cartoon-handdrawn
- cartoon-sketch

日系漫画
3D形态
艺术形态
手绘
素描

```python
from modelscope.hub.snapshot_download import snapshot_download
import argparse


def process(args):
    style = args.style
    print('download %s model'%style)
    if style == "anime":
        model_dir = snapshot_download('damo/cv_unet_person-image-cartoon_compound-models', cache_dir='.')

    elif style == "3d":
        model_dir = snapshot_download('damo/cv_unet_person-image-cartoon-3d_compound-models', cache_dir='.')

    elif style == "handdrawn":
        model_dir = snapshot_download('damo/cv_unet_person-image-cartoon-handdrawn_compound-models', cache_dir='.')

    elif style == "sketch":
        model_dir = snapshot_download('damo/cv_unet_person-image-cartoon-sketch_compound-models', cache_dir='.')

    elif style == "artstyle":
        model_dir = snapshot_download('damo/cv_unet_person-image-cartoon-artstyle_compound-models', cache_dir='.')

    else:
        print('no such style %s'% style)
```

模型内容:

名称

- description
- .msc
- alpha.jpg → 全局mask, 规整人脸的位置
- cartoon_bg.pb → 背景转换
- cartoon_h.pb → 头部转换
- configuration.json
- detector.pb → 检测人脸和人脸关键点
- keypoints.pb → 检测人脸和人脸关键点
- README.md

步骤 1 先整体转换
步骤 2 找到人脸并校正、切割 单独进行人脸转换
步骤 3 将两个结果融合

所有的细节都在 source/cartoonize.py

注意
一点小bug的修改

```
class Cartoonizer():
    def __init__(self, dataroot):

        self.facer = FaceAna(dataroot)
        self.sess_head = self.load_sess(
            os.path.join(dataroot, 'cartoon_h.pb'), 'model_head')
        self.sess_bg = self.load_sess(
            os.path.join(dataroot, 'cartoon_bg.pb'), 'model_bg')
```

需要安装tensorflow

加载风格列表

```python
import cv2
from source.cartoonize import Cartoonizer
import os
import numpy as np


def get_model_list(model_dir):
    list_models = []
    m_dirs = os.listdir(model_dir)
    for dir in m_dirs:
        path_model = os.path.join(model_dir,dir)
        list_models.append(path_model)
    return list_models
```

```python
if __name__ == '__main__':

    file_img = "3474.jpg"
    i=0
    list_models = get_model_list("models")
    algo =Cartoonizer(list_models[i])

    img = cv2.imread(file_img)[...,::-1]
    result = algo.cartoonize(img)
    result_out = np.array(result,dtype=np.uint8)
    # cv2.namedWindow("out", cv2.WINDOW_NORMAL or cv2.WINDOW_KEEPRATIO or cv2.WINDOW_GUI_NORMAL)
    # cv2.imshow("input",cv2.imread('input.png'))
    # cv2.imshow("out",result)
    # cv2.waitKey(0)
    file_img_out = os.path.split(list_models[i])[-1]+"_out_"+file_img
    cv2.imwrite(file_img_out,result_out)
```