

学号	姓名	论文规范性 (10)	问题分析与调研 (30)	方案创新性 (20)	实验结果分析与讨论 (40)	结课论文总成绩 (100)
21301065	韩卓衍	8	23	16	34	81

图像风格迁移，使用预训练模型，偏重于计算机视觉

计算机图形学课程论文

图像风格迁移技术研究

作者：韩卓衍

老师：吴雨婷

北京交通大学

2024 年 6 月

摘要

摘要：图像风格迁移技术，作为计算机图形学领域的重要分支，旨在将一幅图像的艺术风格转移到另一幅图像的内容上，为艺术创作和图像处理领域带来了前所未有的可能性。该技术通过深度学习模型学习并提取图像的风格和内容特征，进而实现风格与内容的融合，生成具有新颖视觉效果的作品。本文实验在飞桨上基于 PaddleHub 的图像风格迁移模型 MSGNet 进行迁移学习，深入探讨了图像风格迁移的实现方法，以期实现高质量风格迁移效果。

关键词：图像风格迁移；深度学习；MSGNet

目录

摘要	ii
1 引言	1
1.1 研究背景和意义	1
1.2 国内外研究现状	1
1.3 图像迁移中的难题	1
1.4 相关工作介绍	2
2 实验设计	3
2.1 实验的软件环境	3
2.2 数据集的选择	3
2.3 模型的选择	3
2.4 数据处理	4
2.5 模型训练	5
3 实验结果与分析	7
3.1 实验结果	7
3.2 实验结果分析	8
4 总结与展望	9
4.1 总结	9
4.2 展望	9
参考文献	10

1 引言

图像风格迁移是计算机图形学分支领域中的研究热点和难点,在很多方面有着广泛的应用和研究价值。本文主要在飞桨上基于 PaddleHub 的图像风格迁移模型 MSGNet 进行迁移学习,旨在实现高质量的图像风格迁移效果。

1.1 研究背景和意义

在数字化时代的浪潮下,计算机图形学和深度学习技术的飞速发展极大地推动了艺术创作和图像处理领域的创新。图像风格迁移是将一幅图像的内容与另一幅图像的风格相结合,合成出新的图像。即:将图像 A 的内容和图像 B 的风格进行融合,摒弃 A 的风格和 B 的内容,生成新的图像 C^[1]。这个最新生成的图像 C 既体现出图像 A 的特征,又涵盖了图像 B 的内容。这种技术不仅为艺术家提供了更为广阔的创作空间,还为图像处理和视觉艺术带来了无限可能,使得人们能够以全新的方式欣赏和创造艺术作品。

1.2 国内外研究现状

目前国内外广泛应用的图像风格迁移方法有多种,根据风格迁移的实现方法和理论基础的不同可划分为:传统的图像风格迁移与基于神经网络的图像风格迁移。其中传统方法侧重于人工设计特征提取和数学优化,而基于神经网络的方法则侧重于端到端的学习,以数据训练为主^[2]。图像风格迁移技术的核心在于如何有效地提取和融合图像的风格与内容特征。传统的图像处理方法往往难以同时满足这两个方面的需求,而深度学习模型的强大学习能力则为解决这一问题提供了新的思路。通过构建深度神经网络,模型能够学习到图像中的丰富信息,包括色彩、纹理、结构等风格特征,以及图像内容的语义信息。近年来,随着深度学习技术的不断发展,图像风格迁移技术也取得了显著的进步。特别是基于生成对抗网络和神经风格迁移的方法,已经能够生成具有高质量风格迁移效果的作品。然而,这些方法往往存在计算量大、训练时间长等问题,限制了其在实际应用中的推广。

1.3 图像风格迁移中的难题

在图像风格迁移过程中,如何平衡风格迁移的强度和内容保留的完整度是一个重要且复杂的问题。过度的风格迁移可能会导致内容信息的丢失,而风格迁移不足则可能无法达到预期的艺术效果。因此,如何设计有效的算法和模型,以实现风格与内容的最佳平衡,是图像风格迁移技术需要解决的关键问题。而在深度学习应用到图像风格迁移中,需要大量的高质量数据进行训练,数据集的多样性和质量直接影响模型的性能和泛化能力,如何找到合适的数据集也是一个难题。

在复杂的神经网络中，通常需要大量的计算资源和时间，提高算法效率，减少计算时间，同时保持迁移质量，是一个持续的挑战。

1.4 相关工作介绍

本文的主要工作是在飞桨上用 PaddleHub 的图像风格迁移模型 MSGNet 对开源的 MiniCOCO 数据集进行训练，将自己选择的风格应用到其他图像上，以实现高质量的图像风格迁移。

2 实验设计

本章主要对本次实验的进行详细说明。首先说明了软件环境，之后又解释了数据集来源和模型选择，最后再说明对数据的处理和训练过程。

2.1 实验的软件环境

在训练图像迁移风格模型的过程中，需要进行极大量的计算和时间，要求较高的算力资源，因而本次的软件环境采用百度自主研发的深度学习平台——飞桨。飞桨是一个集深度学习核心框架、基础模型库、端到端开发套件和丰富工具组件于一体的高效、灵活、易用的平台。通过优化算法和模型结构，飞桨生态特色模型能够在保持准确性的前提下，提高计算速度和效率。

2.2 数据集的选择

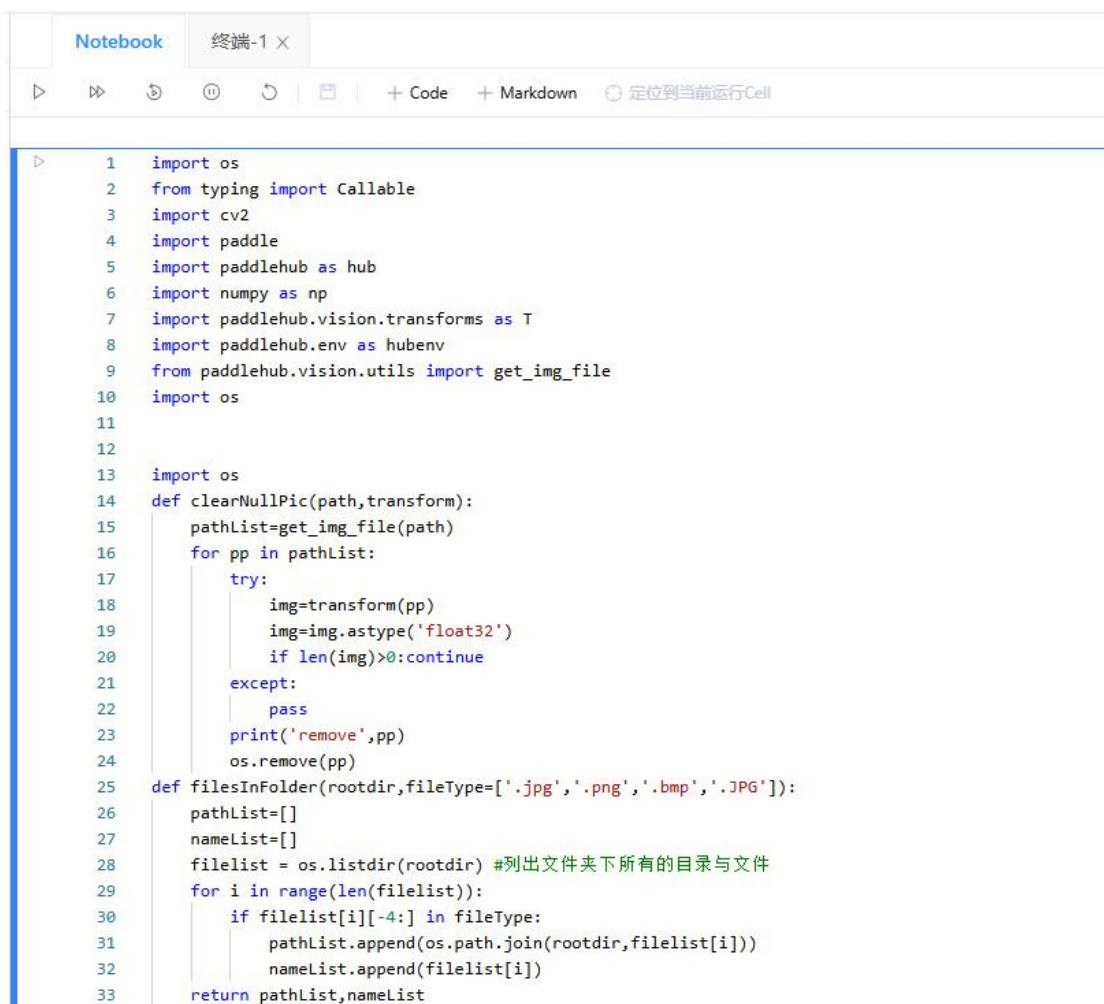
本文采用的 MiniCOCO 数据集来源于 COCO 2014 数据集，其中训练集 2000 张图像，测试集 200 张图像，包含了 21 种不同风格的图片。COCO 数据集是一个大型的图像识别数据集，用于对象检测、分割等任务。数据集的多样性与质量直接影响到图像迁移模型的泛化能力，而 MiniCOCO 数据集作为 COCO 数据集的一个子集，具有轻量级、快速实验、数据多样性和与 COCO 数据集兼容等优点，满足了本次训练模型的数据要求。

2.3 模型的选择

本次模型采用了基于 PaddleHub 的图像迁移模型 MSGNet。MSGNet 模型是由飞桨深度学习平台提供的一种预训练模型，专为图像风格迁移任务设计。其它的传统图像迁移模型有 NST、FastNeuralStyleTransfer 等等。而 NST 方法计算成本高，需要较长的时间和大量的计算资源来进行风格迁移，尤其是在处理高分辨率图像时；FastNeuralStyleTransfer 虽然速度快，但每个训练好的模型只能迁移一种特定的风格，要迁移不同风格需要为每种风格训练一个单独的模型。而 MSGNet 模型在进行图像风格迁移时具有较高的计算速度，能够在短时间内生成高质量的迁移图像；且 MSGNet 克服了 FastNeuralStyleTransfer 灵活性差的缺点，可以通过少量微调适应多种风格；更重要的是，MSGNet 具有良好的泛化能力，能够在不同类型的图像上表现出一致且高质量的迁移效果，适应多样化的应用场景。

2.4 数据处理

由于 MiniCOCO 的有些图片打开会有问题，需要过滤。首先遍历路径下的所有图像文件，尝试加载并转换图像，如果失败则删除该图像文件。接着自定义一个 MyMiniCOCO 类，接受一个转换函数、一个模式（‘train’ 或 ‘test’）、一个风格文件夹和一个风格图片作为参数，如果指定了 ‘styleImg’ 则加载该特定风格图像，否则加载风格文件夹中的所有风格图像。自定义类中的 ‘getImg’ 方法通过索引并转换图像，确保返回非空图像；‘_getitem_’ 方法中，先获取内容图像索引对应的风格图像，并进行预处理；‘_len_’ 方法返回数据集的图像数量，用于迭代训练。使用 Resize 函数将图片大小调整为 256x256。



```
1 import os
2 from typing import Callable
3 import cv2
4 import paddle
5 import paddlehub as hub
6 import numpy as np
7 import paddlehub.vision.transforms as T
8 import paddlehub.env as hubenv
9 from paddlehub.vision.utils import get_img_file
10 import os
11
12
13 import os
14 def clearNullPic(path,transform):
15     pathList=get_img_file(path)
16     for pp in pathList:
17         try:
18             img=transform(pp)
19             img=img.astype('float32')
20             if len(img)>0:continue
21         except:
22             pass
23         print('remove',pp)
24         os.remove(pp)
25 def filesInFolder(rootdir,fileType=['.jpg','.png','.bmp','.JPG']):
26     pathList=[]
27     nameList=[]
28     filelist = os.listdir(rootdir) #列出文件夹下所有的目录与文件
29     for i in range(len(filelist)):
30         if filelist[i][-4:] in fileType:
31             pathList.append(os.path.join(rootdir,filelist[i]))
32             nameList.append(filelist[i])
33     return pathList,nameList
```

图 2-1 必要库导入代码

```

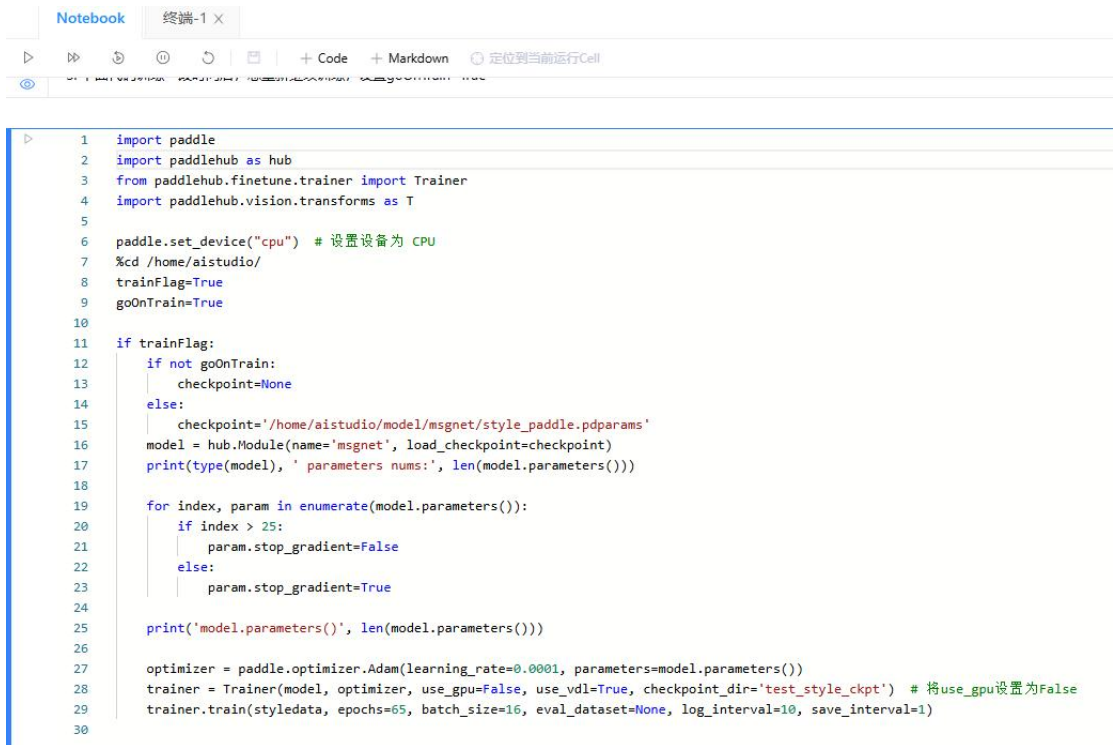
34 class MyMiniCOCO(paddle.io.Dataset):
35
36     def __init__(self, transform: Callable, mode: str = 'train',
37                 data1path='/home/aistudio/data', styleFolder='style', styleImg=None):
38         self.mode = mode
39         self.transform = transform
40         if self.mode == 'train':
41             self.file = 'train'
42         elif self.mode == 'test':
43             self.file = 'test'
44         if styleImg:
45             self.style_file = os.path.join(data1path, styleFolder, styleImg)
46             self.style=[self.style_file]
47         else:
48             self.style_file = os.path.join(data1path, styleFolder)
49             self.style = get_img_file(self.style_file)
50         self.file = os.path.join(data1path, 'minicoco', self.file)
51         self.data = get_img_file(self.file)
52         assert (len(self.style)>0 and len(self.data)>0)
53         print('self.data', len(self.data))
54         print('self.style', len(self.style))
55     def getImg(self, group, idx):
56         im=[]
57         ii=idx
58         while len(im)==0:
59             try:
60                 im = self.transform(group[ii])
61             except :
62                 print('v', len(group), ii)
63             ii-=1
64         return im
65     def __getitem__(self, idx: int) -> np.ndarray:
66         im = self.getImg(self.data, idx)
67         im = im.astype('float32')
68         style_idx = idx % len(self.style)
69         style = self.getImg(self.style, style_idx)
70         style = style.astype('float32')
71         return im, style
72     def __len__(self):
73         return len(self.data)

```

图 2-2 自定义 MyMiniCOCO 类代码

2.5 模型训练

首先将设备设置为 CPU，载入基于 PaddlePaddle 的 MSGNet 模型。如果不继续训练已有模型，则不加载检查点，如果继续训练，则加载指定路径的模型检查点。通过遍历模型的参数，设置前 25 个参数的梯度计算为不可用，后续参数的梯度计算为可用。使用 Adam 优化器，并设置学习率为 0.0001，使用 CPU 进行训练，epoch 为 65，batch_size 为 16，每 10 次 batch 记录日志，每 1 个 epoch 保存模型检查点。



```
1 import paddle
2 import paddlehub as hub
3 from paddlehub.finetune.trainer import Trainer
4 import paddlehub.vision.transforms as T
5
6 paddle.set_device("cpu") # 设置设备为 CPU
7 %cd /home/aistudio/
8 trainFlag=True
9 goOnTrain=True
10
11 if trainFlag:
12     if not goOnTrain:
13         checkpoint=None
14     else:
15         checkpoint='/home/aistudio/model/msgnet/style_paddle.pdparams'
16     model = hub.Module(name='msgnet', load_checkpoint=checkpoint)
17     print(type(model), ' parameters nums:', len(model.parameters()))
18
19     for index, param in enumerate(model.parameters()):
20         if index > 25:
21             param.stop_gradient=False
22         else:
23             param.stop_gradient=True
24
25     print('model.parameters()', len(model.parameters()))
26
27     optimizer = paddle.optimizer.Adam(learning_rate=0.0001, parameters=model.parameters())
28     trainer = Trainer(model, optimizer, use_gpu=False, use_vdl=True, checkpoint_dir='test_style_ckpt') # 将use_gpu设置为False
29     trainer.train(styledata, epochs=65, batch_size=16, eval_dataset=None, log_interval=10, save_interval=1)
30
```

图 2-3 模型训练代码

3 实验结果与分析

本次实验使用一张星空图作为风格背景，迁移到一张内容图片上，查看图像风格迁移效果。

3.1 实验结果

指定内容图和风格图路径，使用 OpenCV 库读取内容图像，加载自己训练的 MSGNet 模型并对内容图像进行风格迁移，将结果保存到指定输出目录。将风格迁移后的图像调整回原始图像的尺寸比例。保存风格迁移后的图像到指定路径，并使用 Matplotlib 显示原始图像和风格迁移后的图像。

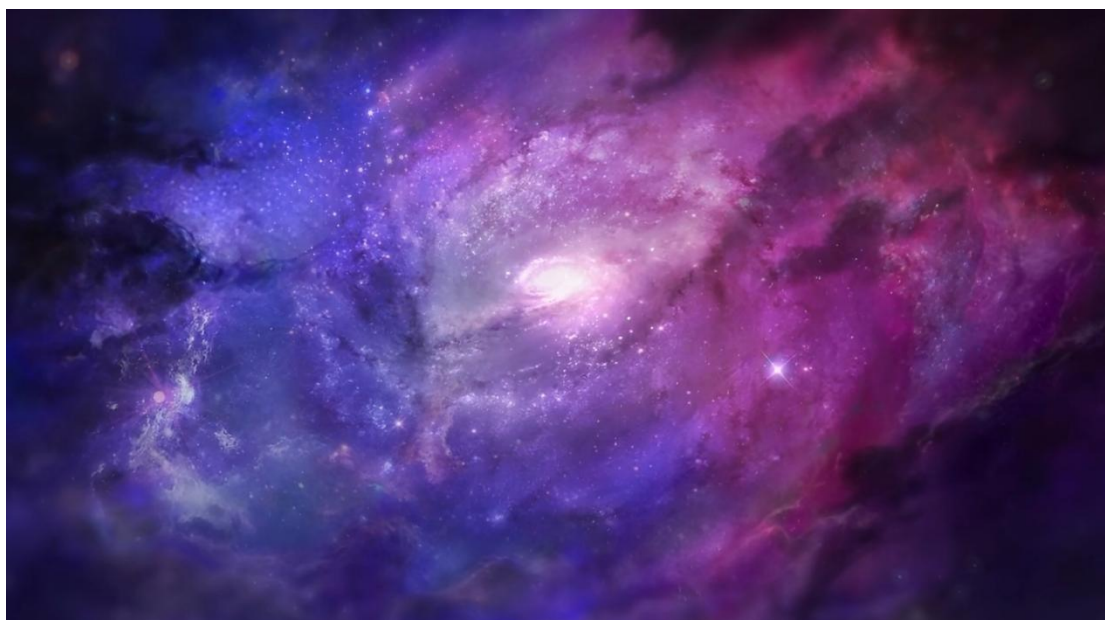


图 3-1 风格图

```
[ ] 1 import paddle
    2 import paddlehub as hub
    3 import cv2
    4 import matplotlib.pyplot as plt
    5 %matplotlib inline
    6 paddle.disable_static()
    7 if __name__ == '__main__':
    8     outputPath='work/result/bjtu.png'
    9     styleImgPath="work/style.jpg"
   10     testImgPath='work/content/bjtu.jpg'
   11     image=cv2.imread(testImgPath)
   12     #
   13     model_ori = hub.Module(name='msgnet')
   14     result_ori=model_ori.predict([image], style=styleImgPath, visualization=True, save_path='style_transfer')
   15     del model_ori
   16     #
   17     model = hub.Module(directory='work/model/msgnet')
   18     result = model.predict([image], style=styleImgPath, visualization=True, save_path='style_transfer')
   19     #由正方形输出拉回原来图像比例
   20     result[0]=cv2.resize(result[0],(image.shape[1],image.shape[0]),3)
   21     cv2.imwrite(outputPath,result[0])
   22     plt.figure(figsize=(10,10))
   23     plt.subplot(1,2,1)
   24     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
   25     plt.title('original image')
   26     plt.subplot(1,2,2)
   27     plt.imshow(result[0][:,:,[2,1,0]])
   28     plt.title('stylized image')
   29     plt.show()
```

load pretrained checkpoint success
load pretrained checkpoint success

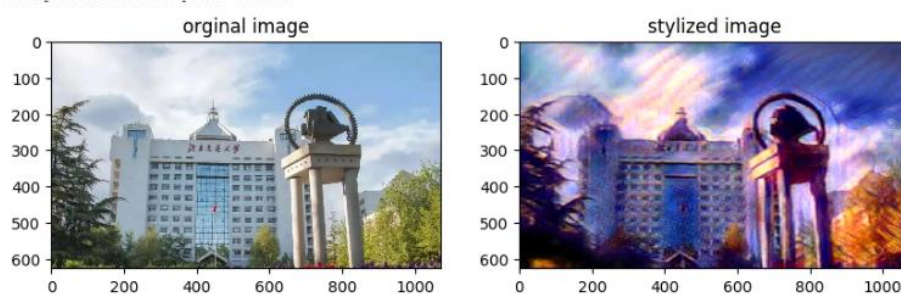


图 3-2 模型调用代码和图像风格迁移结果

3.2 实验结果分析

调用训练好的 MSGNet 模型，使得风格迁移后的图像成功地融合了内容图像的结构和风格图像的艺术元素。星空风格的色调和纹理很好地应用到了原始图像中，创造出一种新的视觉效果。尽管风格迁移在大范围内效果显著，但在细节部分存在一定程度的模糊和信息丢失。其实为了降低计算复杂度和内存消耗，MSGNet 模型通常会将图像缩小到特定的分辨率，这种降采样过程会导致图像中的细节信息丢失，从而影响最终的迁移效果。在后续的优化中，打算在模型训练阶段使用高分辨率的图像，以帮助模型捕捉更多的细节信息。

4 总结与展望

本章对本文在飞桨上基于 PaddleHub 的图像风格迁移模型 MSGNet 的实验进行分析和总结，并对未来的工作进行展望。

4.1 总结

在本研究中，本文使用飞桨平台上的 PaddleHub 进行了基于 MSGNet 模型的图像风格迁移实验。通过将星空图作为风格图迁移到内容图像上，成功实现了风格与内容的融合，生成了具有艺术效果的全新图像。基于 PaddleHub 的 MSGNet 模型在实现图像风格迁移方面具有较高的有效性和灵活性。尽管存在细节模糊和信息丢失的问题，但整体迁移效果显著。

4.2 展望

当前，图像风格迁移技术还需要更多的学习和研究，基于本文的 MSGNet 模型的实验结论，还可以在以下方面进一步尝试：

（1）在训练和生成过程中使用高分辨率图像，以保留更多细节，提升最终生成图像的质量；引入多尺度网络架构，如特征金字塔网络，通过融合不同尺度的特征图，提升细节表现和整体效果。

（2）学习其他图像风格迁移技术和模型，将其他模型的优点整合到 MSGNet 模型中或者使用混合模型，更好地实现图像风格迁移效果。

参考文献

- [1] 王茜. 图像风格迁移技术研究[J]. 吕梁学院学报, 2022, 12 (02) : 37-39.
- [2] 廉露, 田启川, 谭润, 张晓行. 基于神经网络的图像风格迁移研究进展[J]. 计算机工程与应用, 2024, 60 (09) : 30-47