阿里云天池牛年读书会

摄影图像算法与典型实践

分享嘉宾：言有三
阿里云MVP，深度学习算法专家

天池读书会

TIANCHI天池

异步社区
人民邮电出版社

《深度学习之摄影图像处理 核心算法与案例精粹》

从摄影作品的美学评价算法，自动构图算法，过渡到对

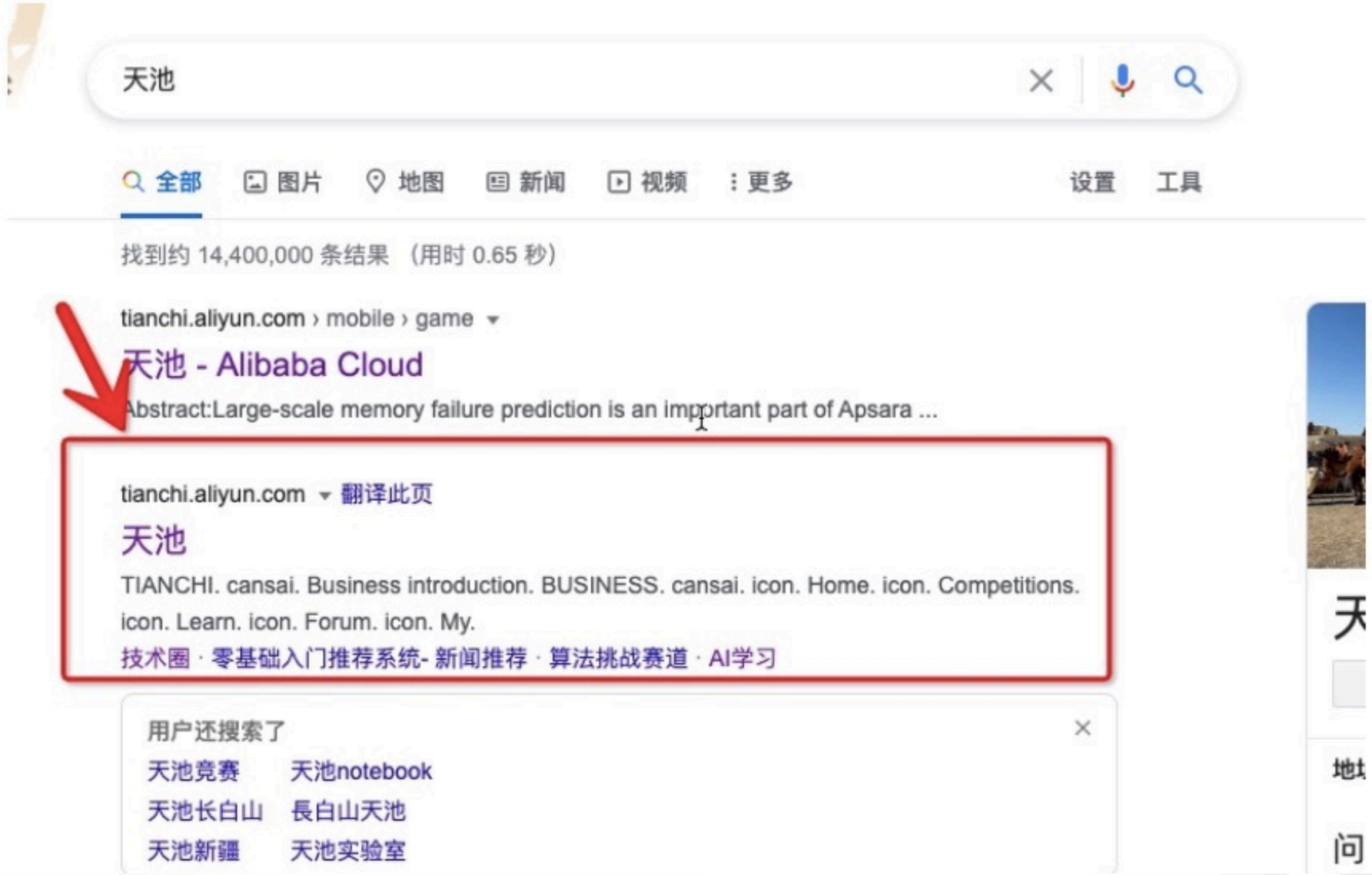
摄影作品的常用编辑技术，内容由浅入深，覆盖了大量应用场景

直播嘉宾：言有三

直播时间：5月25日 20:00

扫码领取读书会配套学习资源

1）首先需要进入天池官网，大家打开浏览器，搜索 天池 ，找到 tianchi.aliyun.com即可访问进入天池官网；



2）在天池官网，将鼠标移到 天池学习 ，即可出现下拉列表，点击 天池读书会 ，即可进入天池读书会的页面。

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

3）在天池读书会页面，你可以对对应的读书会图书进行提问，优秀的提问还有机会获得赠书，还可以点击配套的训练营或者课程资源进入学习，还有点击实践代码获取读书会的项目实践的代码，跟着我一起进行项目实践和代码学习，同时还有很多其他的读书会，大家也可以观看举办过的读书会的回放，或者预约还没开始的读书会。

言有三 本书作者、有三AI创始人

直播主题 《深度学习之摄影图像处理 核心算法与案例精粹》

直播时间 2021年5月25日 20:00

学习资料 深度学习训练营

实践项目 图像风格化实战

提问 | 学习训练营 | 购买地址 | PPT下载 | 实践代码 | 预约直播

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

抽奖提示

言有三 本书作者、有三AI创始人

直播主题 《深度学习之摄影图像处理 核心算法与案例精粹》

直播时间 2021年5月25日 20:00

学习资料 深度学习训练营

实践项目 图像风格化实战

提问 | 学习训练营 | 购买地址 | PPT下载 | 实践代码 | 预约直播

* 1 你的天池昵称?

我们将从所有提问学习者中选择3-5位优秀提问的学习者，赠书一本。

请输入你的回答

* 2 你对以下那本书的内容进行提问? (单选)

- ○ 《人工智能数学基础》
- ○ 《机器学习与Python实践》
- ◉ 《深度学习之摄影图像处理 核心算法与案例精粹》
- ○ 《利用Python进行数据分析(第二版)》
- ○ 《深入浅出图神经网络：GNN原理解析》
- ○ 《？？》等你来推荐

* 3 你的问题描述（越具体越好）？

感谢您的反馈

目录

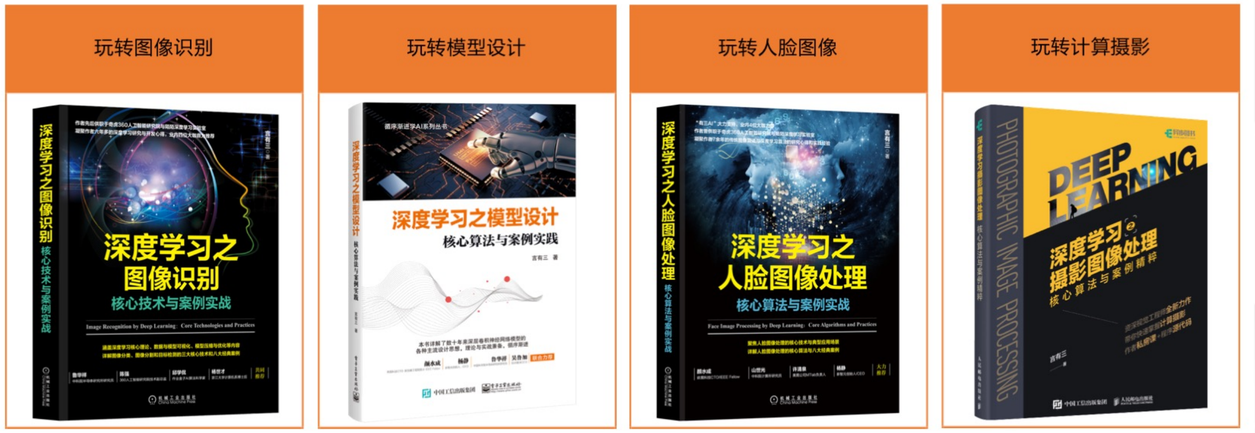
TIANCHI天池

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

作者简介

龙鹏，笔名言有三，先后就读于华中科技大学(2008-2012)，中国科学院(2012-2015)，先后就职于奇虎360人工智能研究院，陌陌科技深度学习实验室，阿里云MVP，华为云MVP，超过7年的计算机视觉从业经验，拥有丰富的传统图像算法和深度学习计算机视觉项目经验。擅长神经网络与深度学习理论，深度学习模型设计与优化，计算机视觉基础领域，AI美学，2D与3D人脸算法，生成对抗网络GAN等领域

运营技术公众号《有三AI》，著有书籍《深度学习之图像识别：核心技术与案例实战》(机械工业出版社2019.4)，《深度学习之模型设计：核心算法与案例实践》(电子工业出版社2020.6)，《深度学习之人脸图像处理：核心算法与案例实战》(机械工业出版社2020.7)，《深度学习之摄影图像处理：核心算法与案例实战》(人民邮电出版社2021.4)。



直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

图书简介

业界首本系统性讲述深度学习摄影图像处理算法的书籍，理论和实践结合

第1章：摄影基础

第2章：图像美学

第3章：自动构图

第4章：图像降噪

第5章：图像对比度与色调增强

第6章：人脸美颜与美妆

第7章：图像去模糊与超分

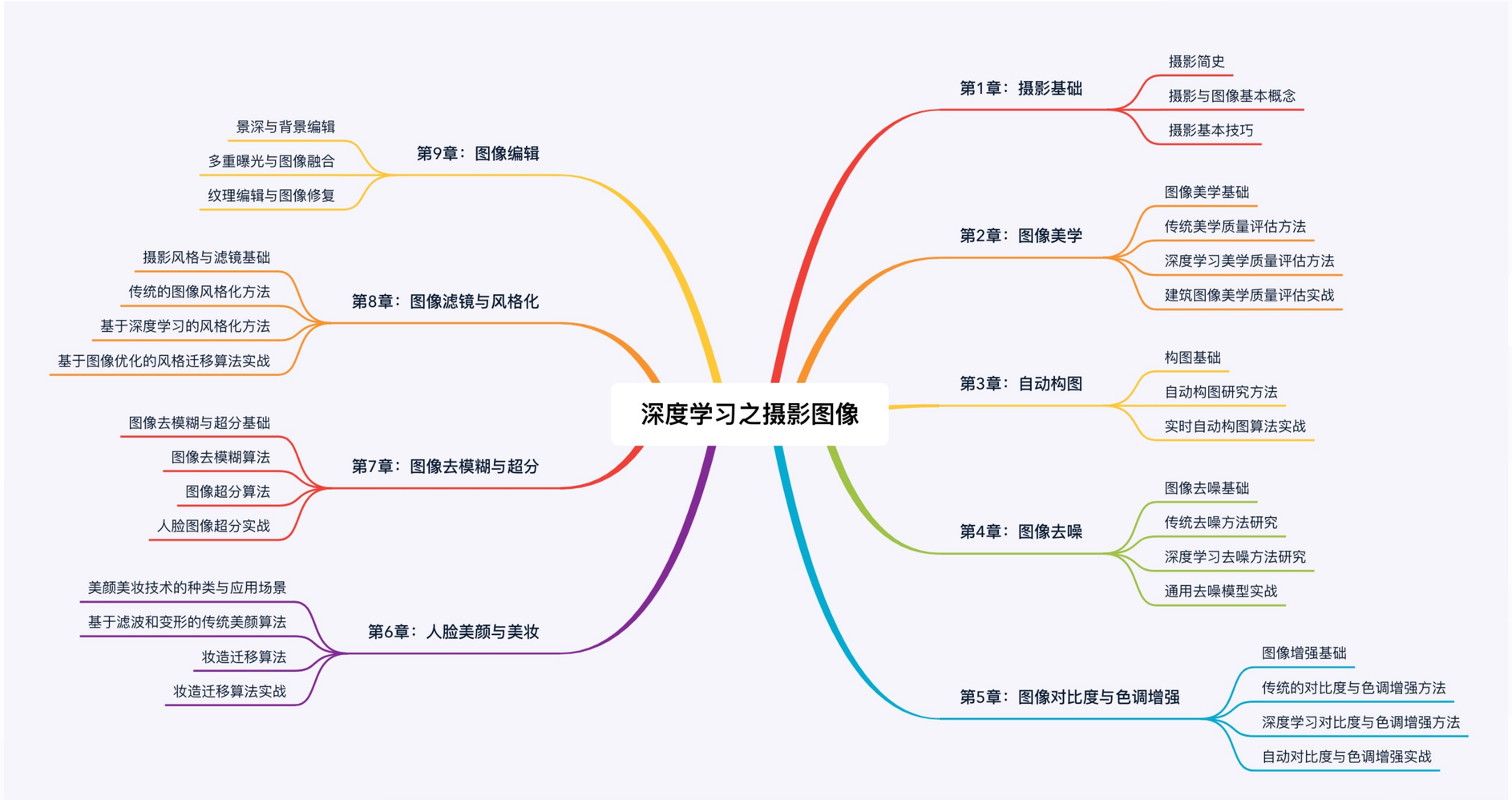
第8章：图像滤镜与风格化

第9章：图像编辑

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

图书内容

目录与案例预览



直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

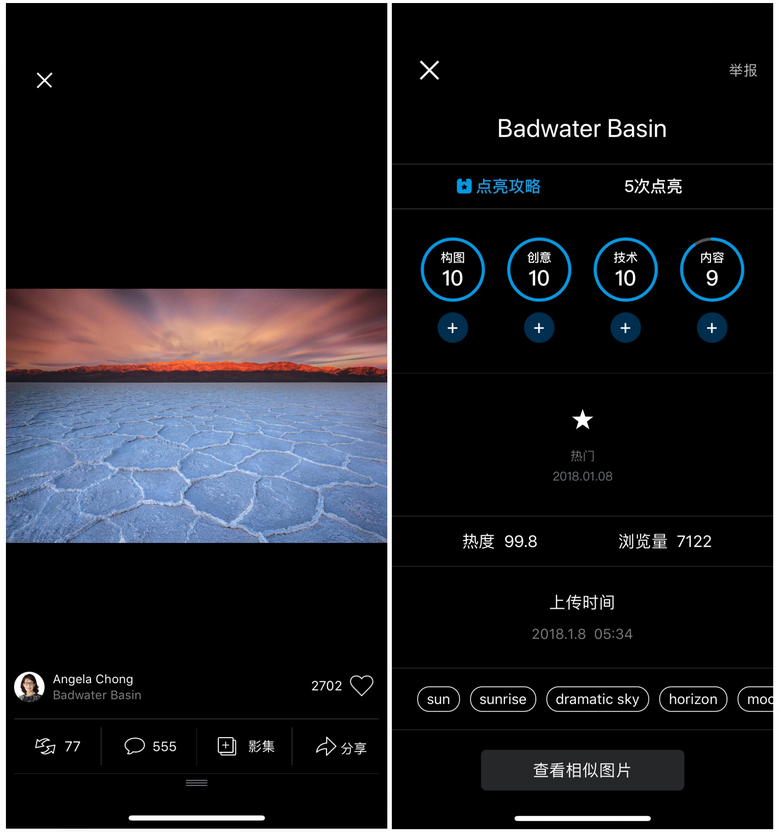
应用领域

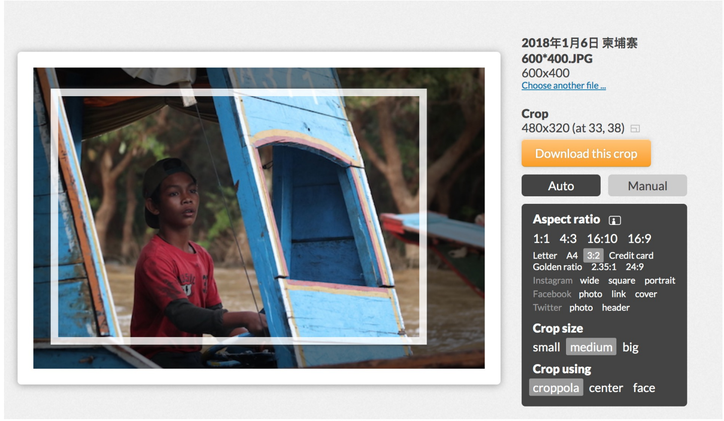
硬件产品

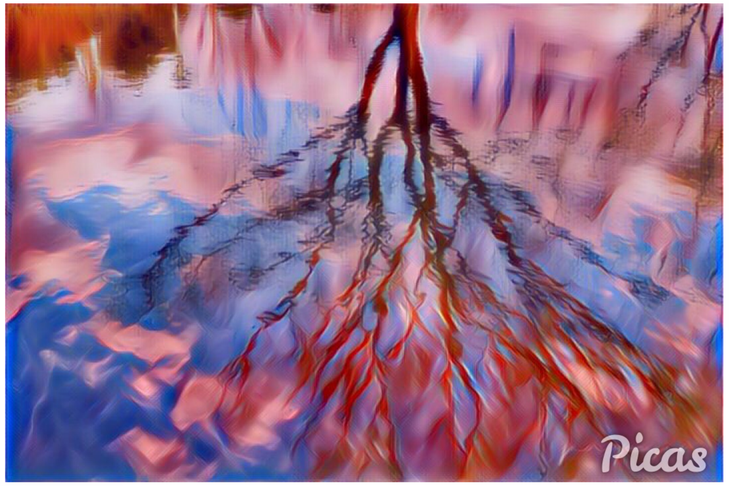
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

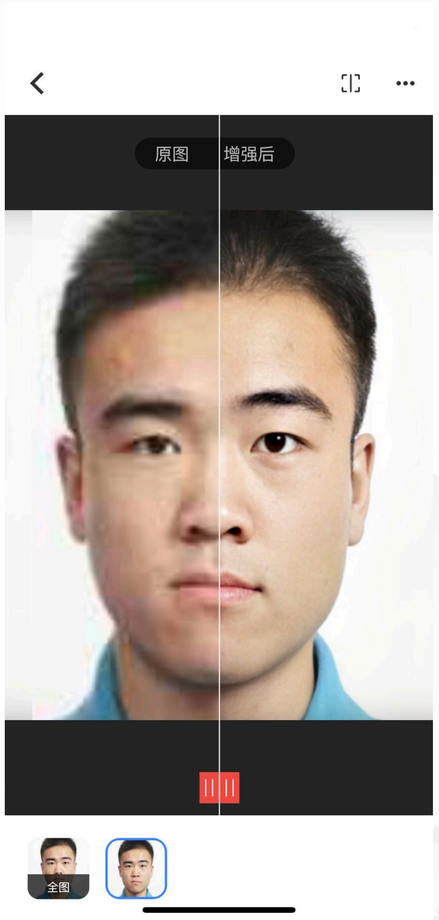
应用领域

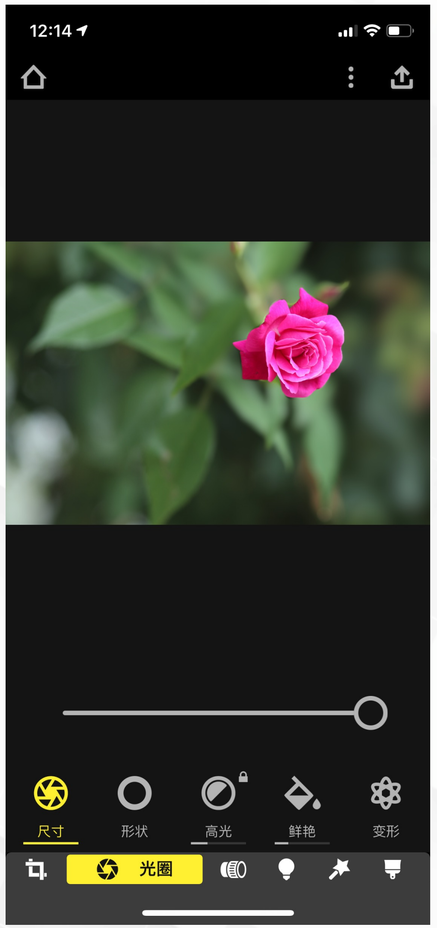
软件产品









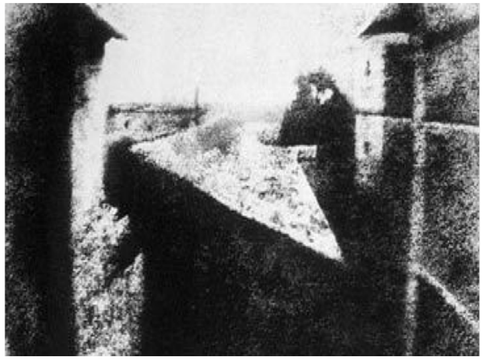
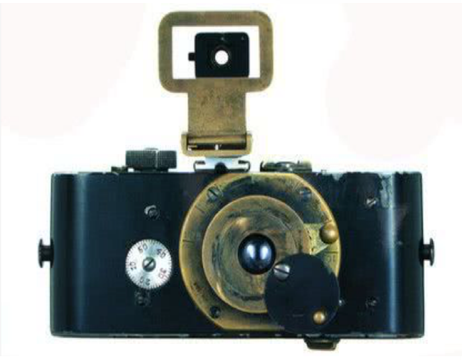


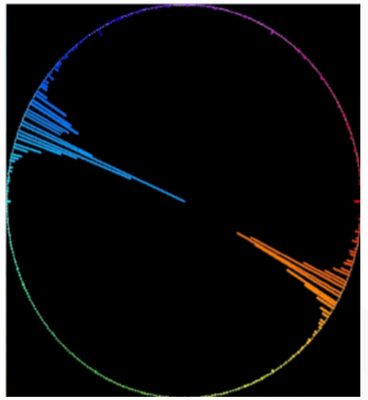
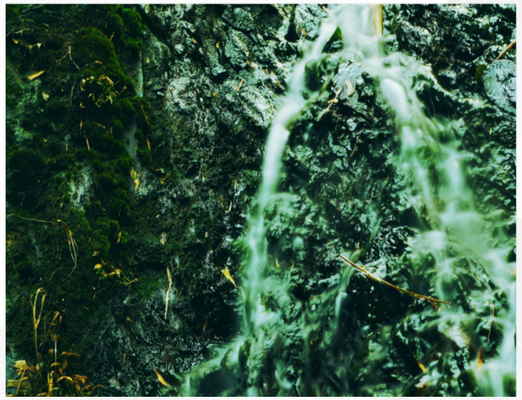
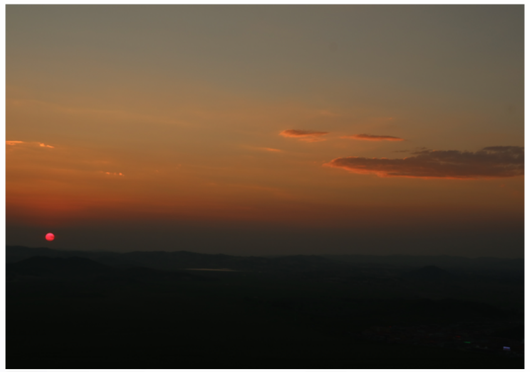
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

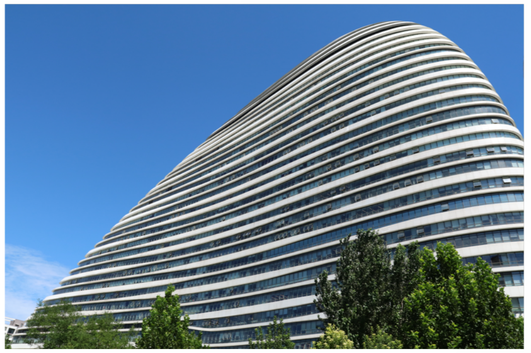
图书内容

第一章：摄影基础

包含摄影简史，图像基础与摄影概念（像素与深度，焦距，光圈，ISO，快门，色温与白平衡，对比度与清晰度等）。



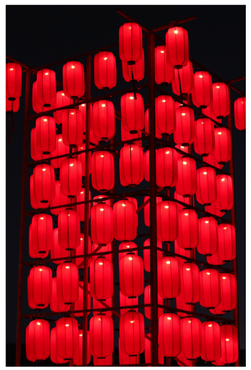

直播相关资料获取及回放查看地址： https://tianchi.aliyun.com/specials/promotion/activity/bookclub

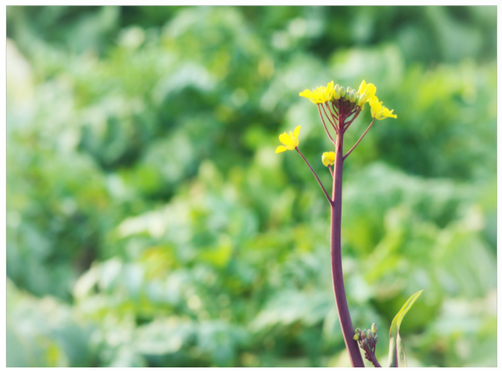
图书内容

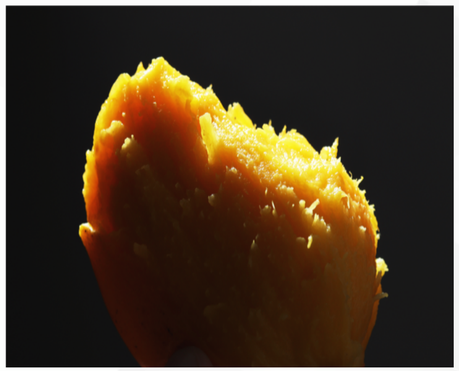
第一章：摄影基础

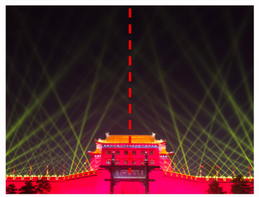
包含摄影技巧(颜色使用，构图方法，用光技巧等)。

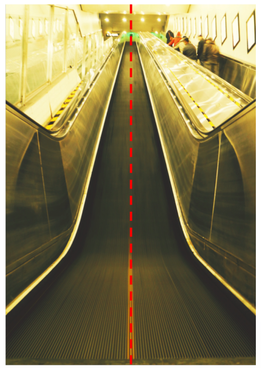
















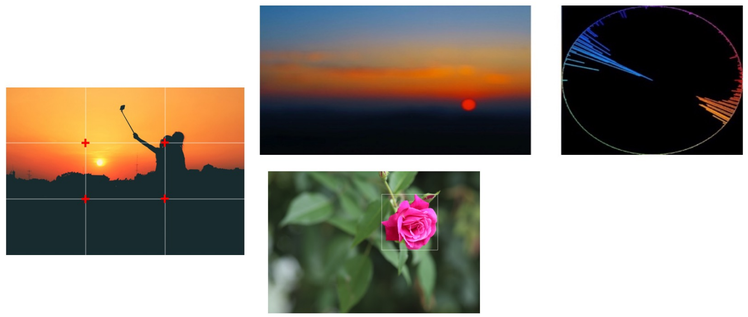


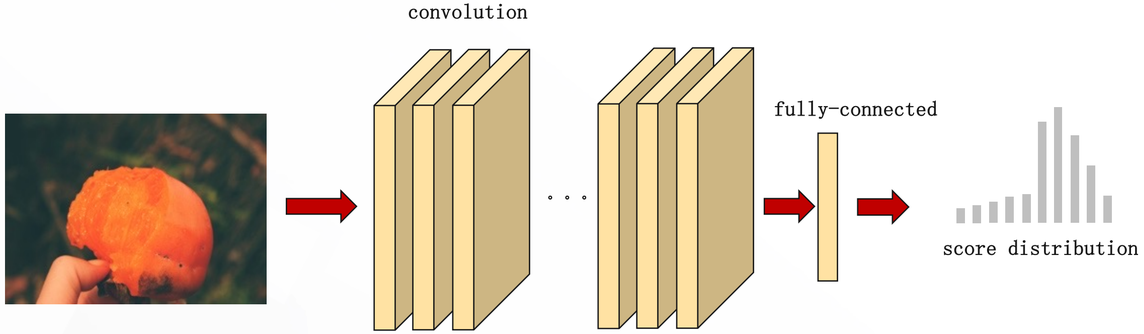

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

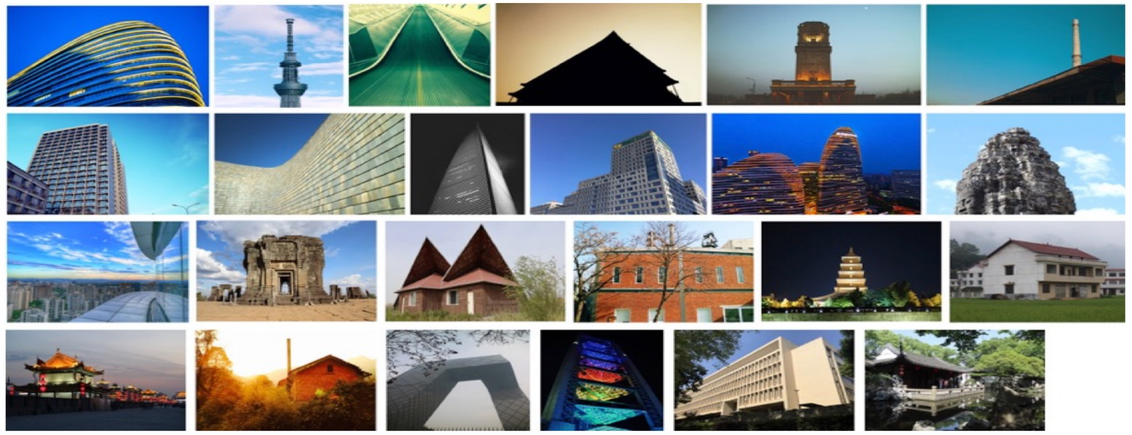
图书内容

第二章：图像美学

包含美学评估的应用场景（检索，构图等），传统的美学评估算法和基于深度学习的评估算法。





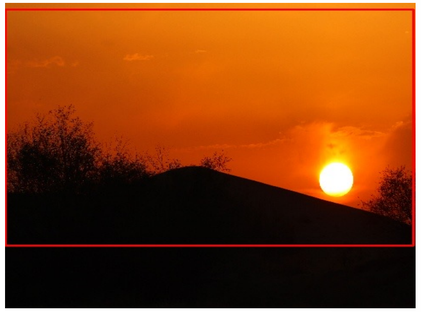
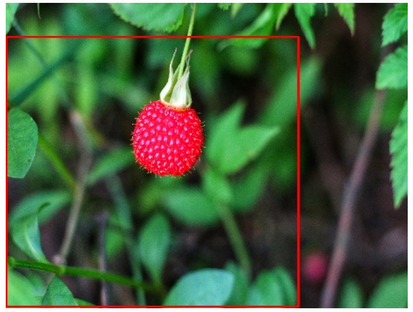


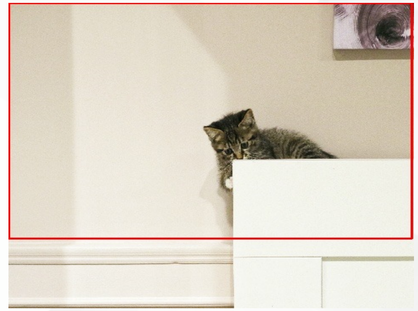
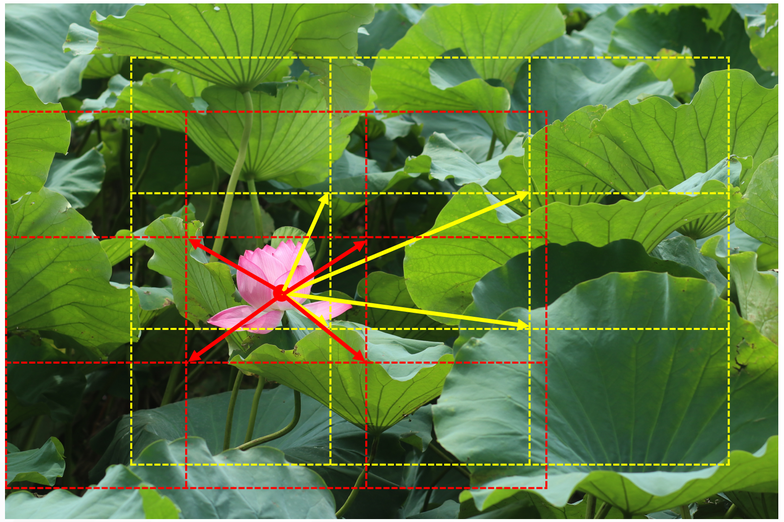
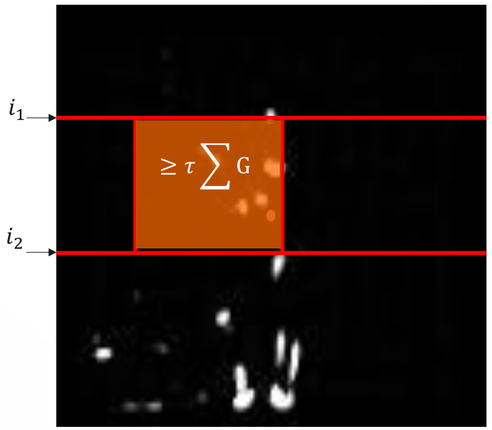
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

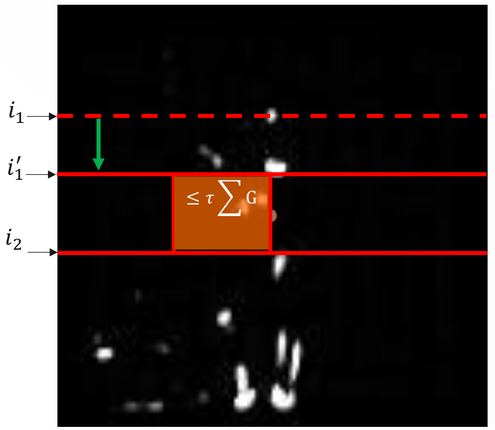
图书内容

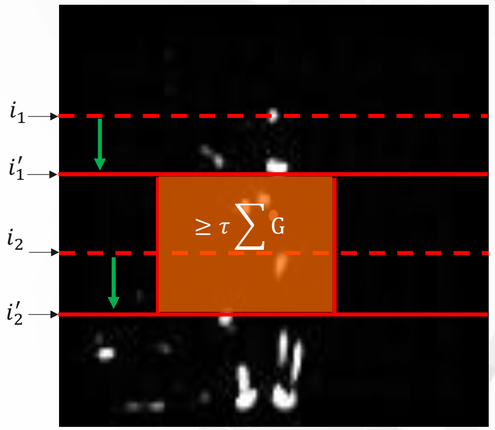
第三章：自动构图

包含自动构图的应用，传统的自动构图算法和基于深度学习的自动构图算法 。



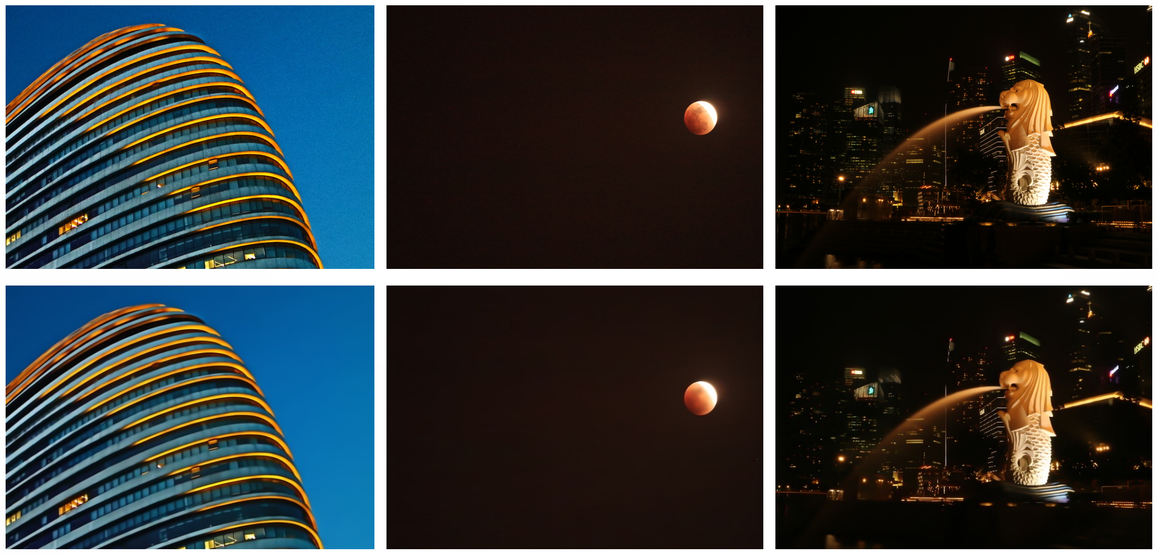


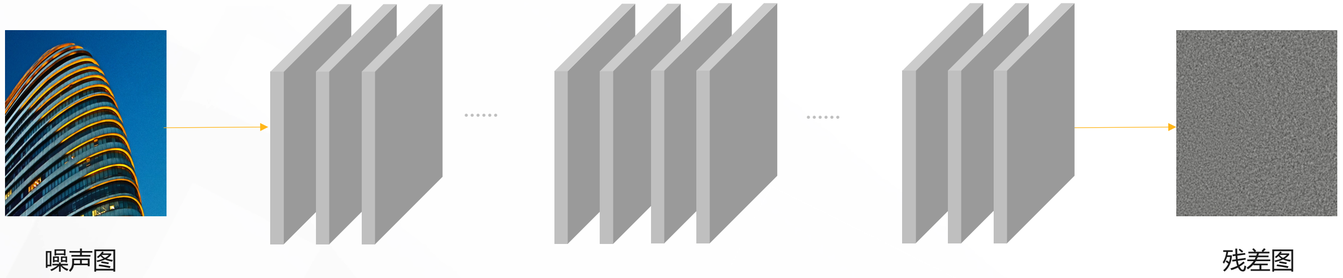
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

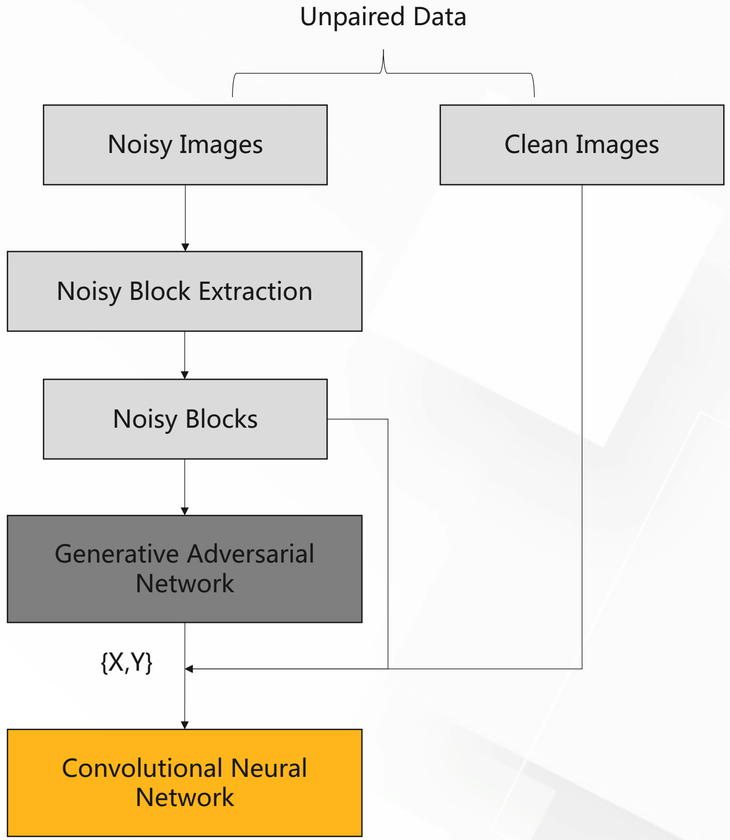
图书内容

第四章：图像降噪

包含噪声的来源和种类，传统的图像降噪算法和基于深度学习的图像降噪算法



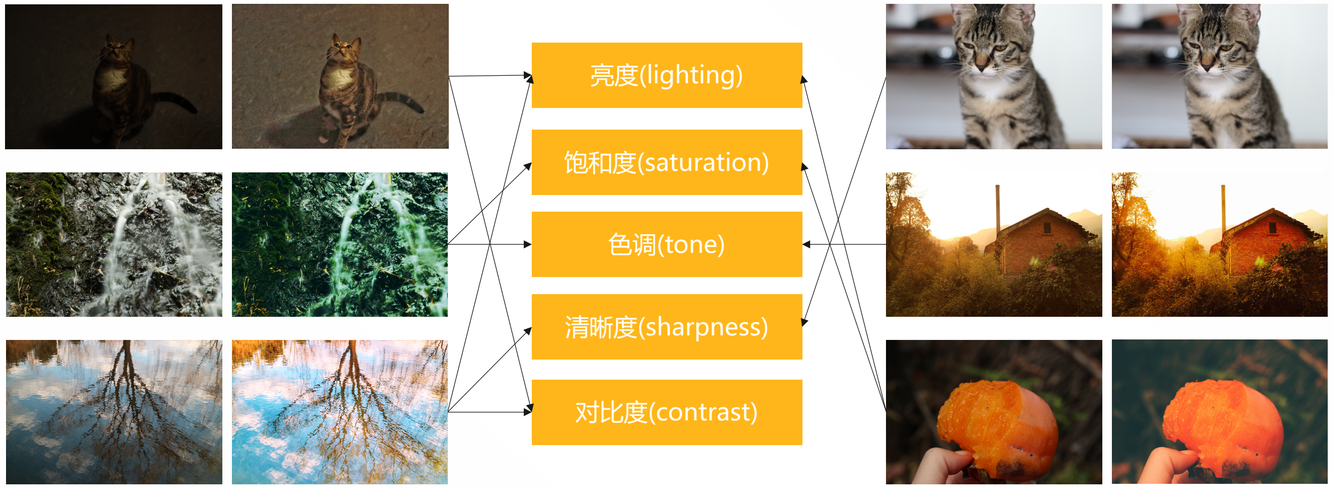


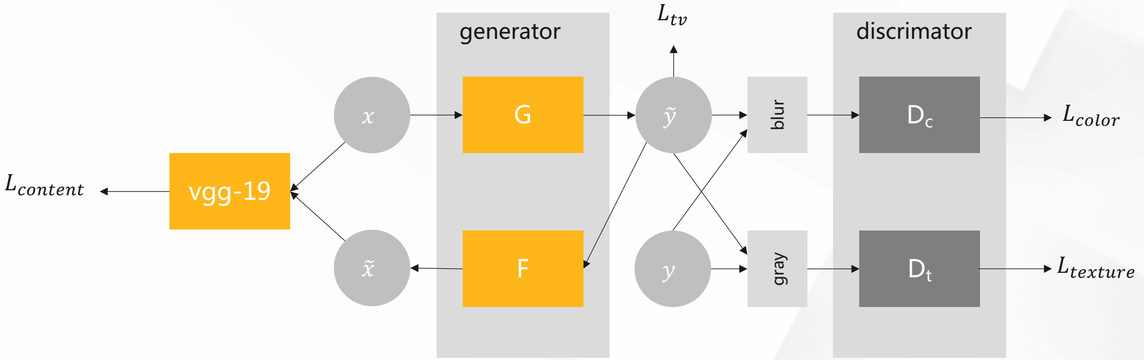


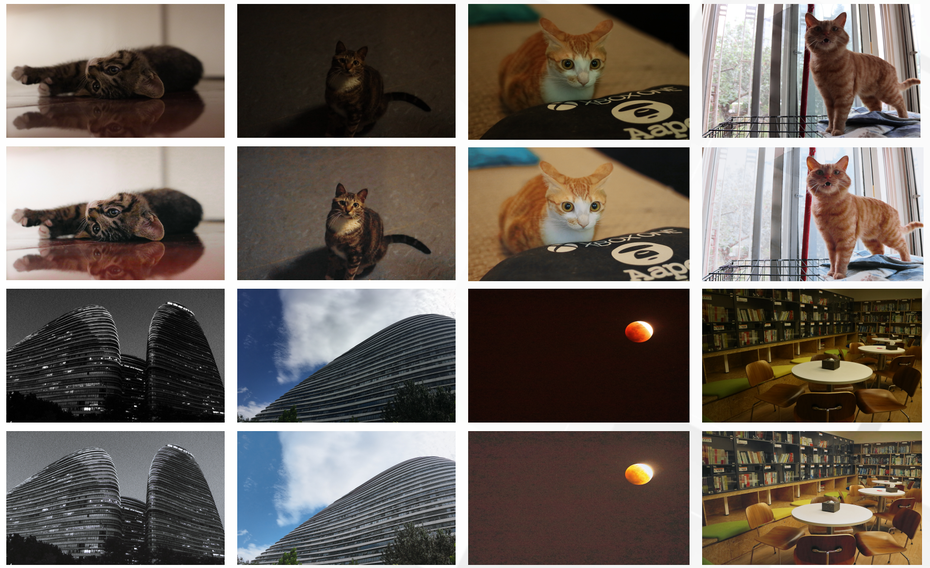
图书内容

第五章：对比度与色调增强

包括摄影中图像对比度和色调增强相关的内容，以及传统的图像增强算法和深度学习图像增强算法





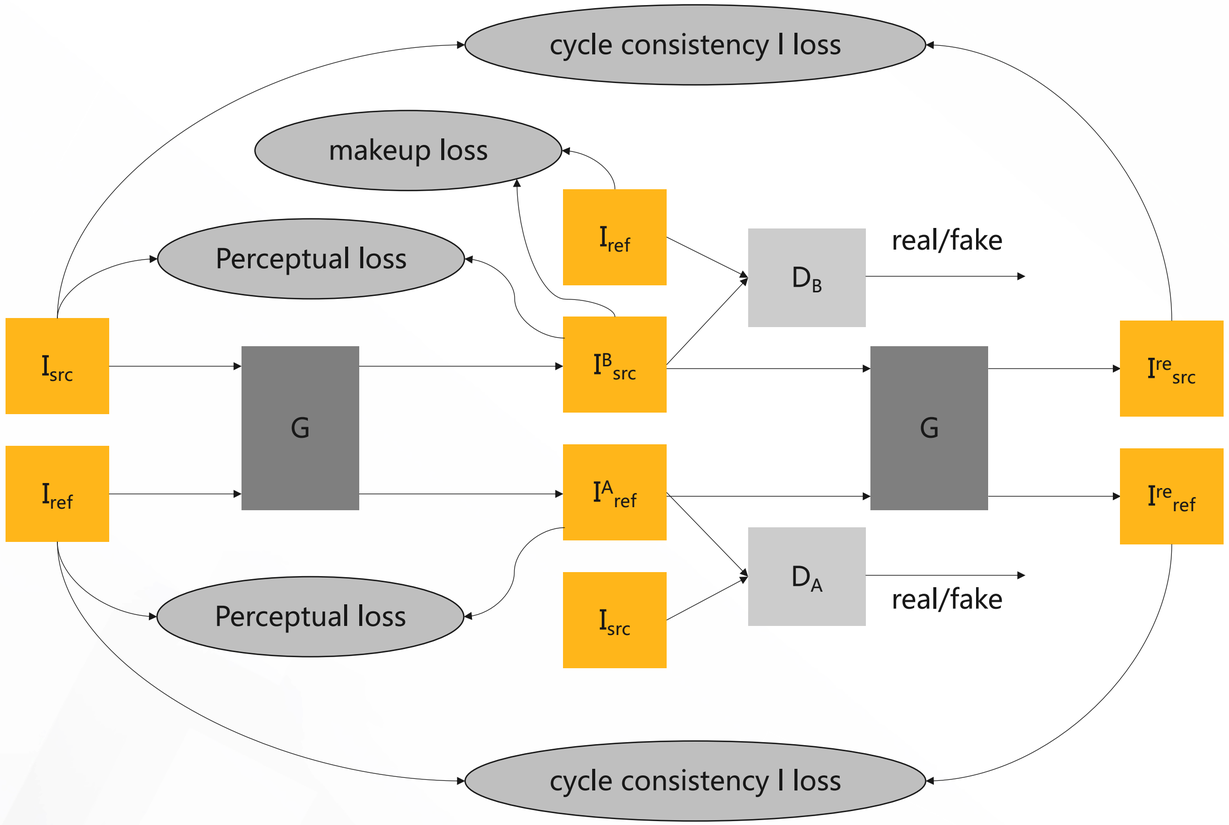


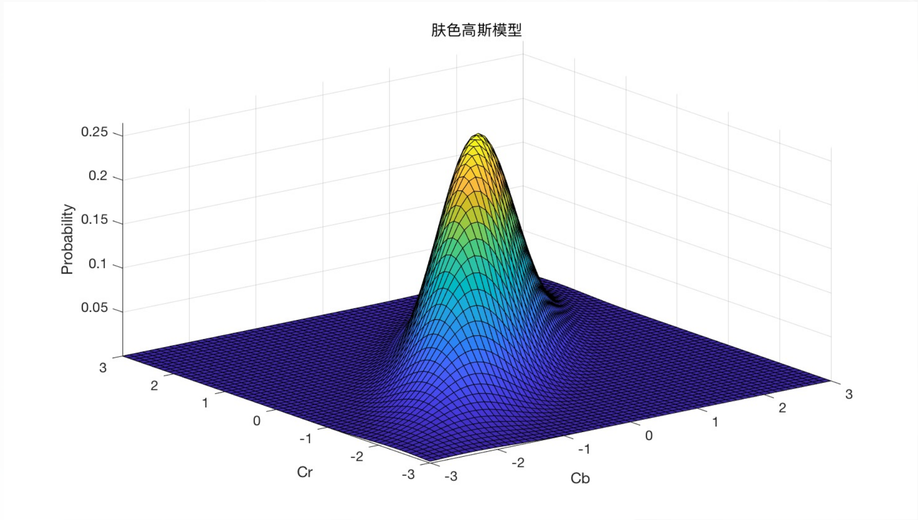
直播相关资料获取及回放查看地址： https://tianchi.aliyun.com/specials/promotion/activity/bookclub

图书内容

第六章：人脸美颜与美妆

包含传统的人脸图像美颜和美妆算法，以及基于深度学习模型的妆造迁移算法



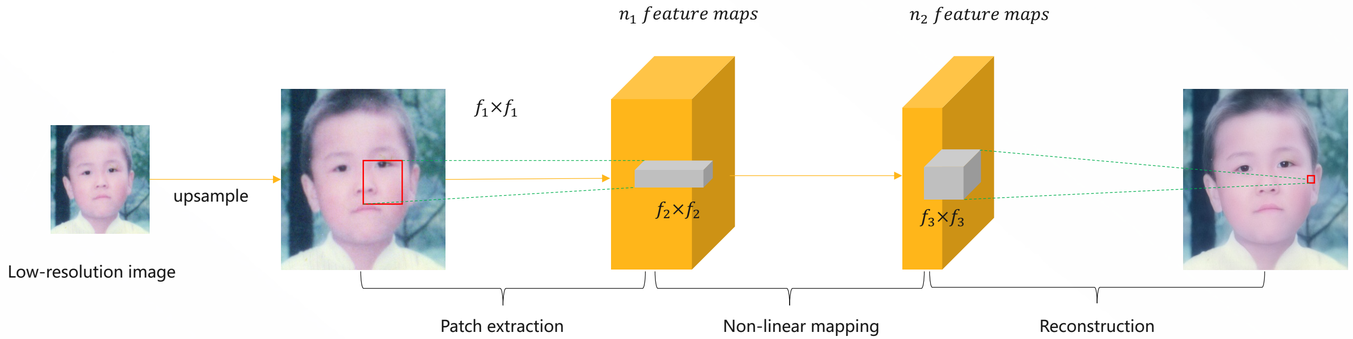


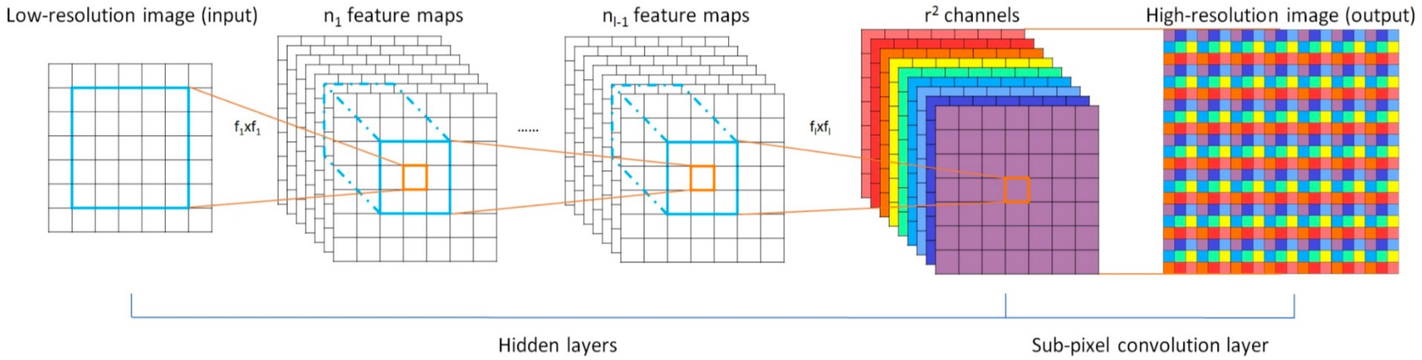
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

图书内容

第七章：图像去模糊与超分

包含图像的超分辨算法和去模糊算法的应用，相应的传统算法和基于深度学习模型的算法



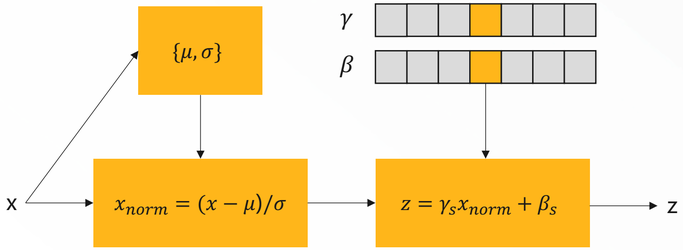


直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

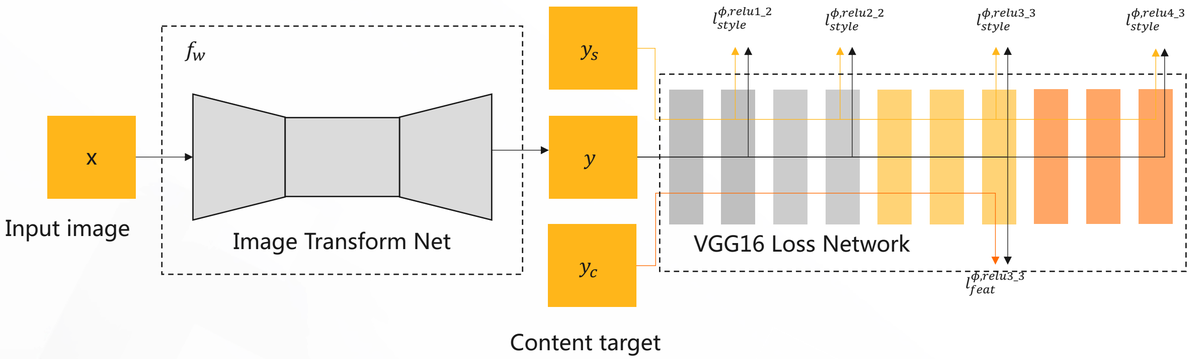
图书内容

第八章：图像滤镜与风格化

包含图像滤镜和风格化的常见类型，以及基于深度学习的图像风格化技术



Style target



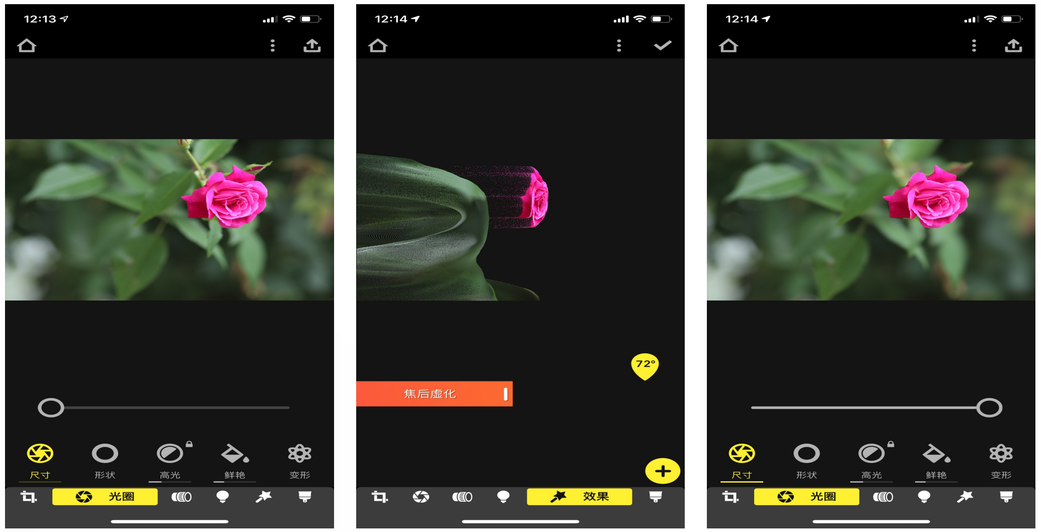
Content target

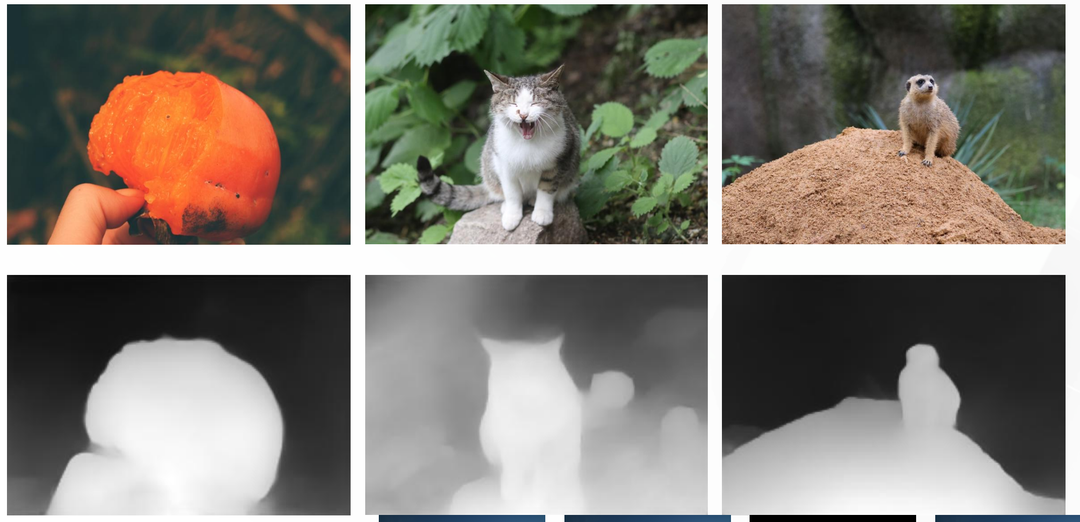
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

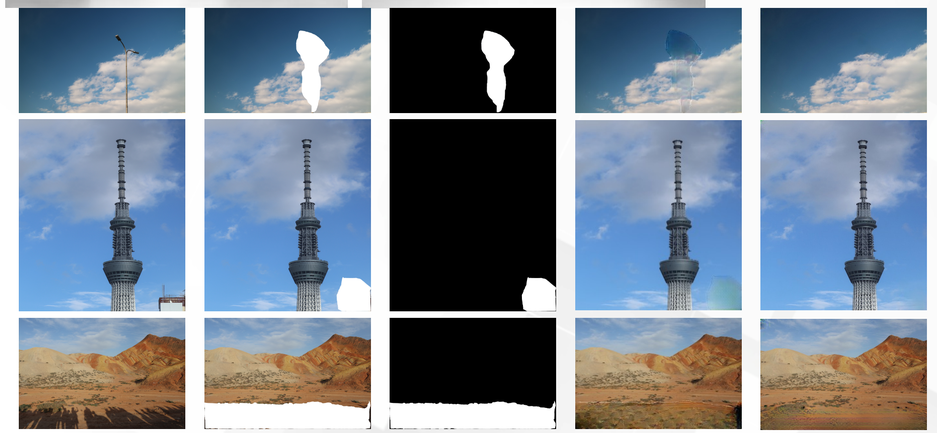
图书内容

第九章：图像编辑

包含摄影图像编辑的各个方向，包括景深的编辑，纹理的编辑以及多幅图像的融合







直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

图书特色

TIANCHI天池

- **循序渐进，内容全面**。本书首先从摄影中的基础概念，尤其是其中涉及到的图像知识开始讲述，然后过渡到摄影与图像交叉的各个领域。从摄影作品的美学评价算法，自动构图算法，过渡到对摄影作品的常用编辑技术，包括噪声去除、对比度增强、色调增强、去模糊以及分辨率提升，然后对当前更先进的人像美颜、图像风格化技术进行了介绍，最后介绍了最有难度的图像景深编辑，纹理编辑，图像融合技术。内容由浅入深，覆盖了大量应用场景，适合系统性进阶学习。

- **传统算法和深度学习算法兼具**。虽然本书名为《深度学习之摄影图像处理》，但是书中每一章都有一定的篇幅在讲述传统图像处理算法，供大家拓展学习。

- **实践充分，由浅入深**。书中内容的章节设置都是先系统阐述理论，然后紧接着选取最具有代表性的内容进行项目实践。

案例讲解

图像风格迁移

任务解读：基于图像的风格迁移
应用场景：社交娱乐等

案例讲解

项目地址：
https://tianchi.aliyun.com/specials/promotion/aicampdl

言有三 本书作者、有三AI创始人

直播主题 《深度学习之摄影图像处理 核心算法与案例精粹》

直播时间 2021年5月25日 20:00

学习资料 深度学习训练营

实践项目 待定

提问 | 学习训练营 | 购买地址 | 预约直播

学习任务

01 Task 1

基于人脸的常见表情识别(1)——深度学习基础知识

学习打卡

点击开启学习

02 Task 2

基于人脸的常见表情识别(2)——数据获取与整理

学习打卡

点击开启学习

03 Task 3

基于人脸的常见表情识别(3)——模型搭建、训练与测试

学习打卡

点击开启学习

Task1.ipynb | Task2.ipynb | Task3.ipynb

Code | Python 3

```
[ ]: #coding:utf8
from __future__ import print_function, division

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
from torch.autograd import Variable
import torchvision
from torchvision import datasets, models, transforms
import time
import os
from tensorboardX import SummaryWriter
import torch.nn.functional as F
import numpy as np

import warnings

warnings.filterwarnings('ignore')

writer = SummaryWriter()

def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        for phase in ['train', 'val']:
            if phase == 'train':
                scheduler.step()
                model.train(True)  # Set model to training mode
```

Task1：理论简介
Task2：模型解读
Task3：模型训练与测试

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

案例讲解

TIANCHI天池

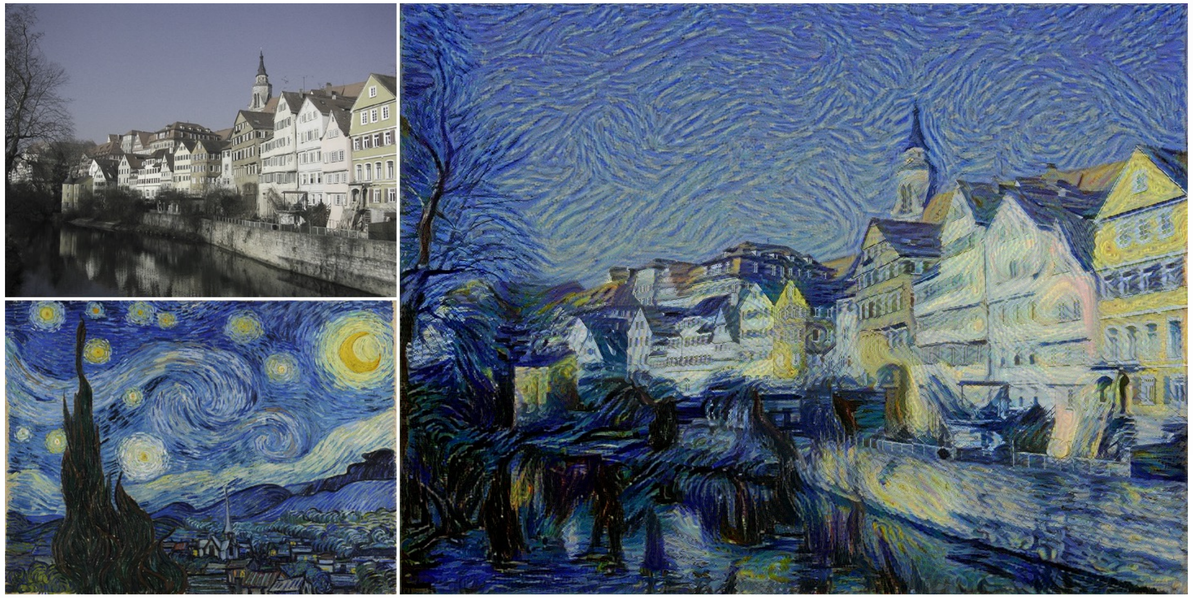
完整的流程

选择开源学习框架 → 风格化原理解读 → 网络模型 → 模型训练 → 更多测试

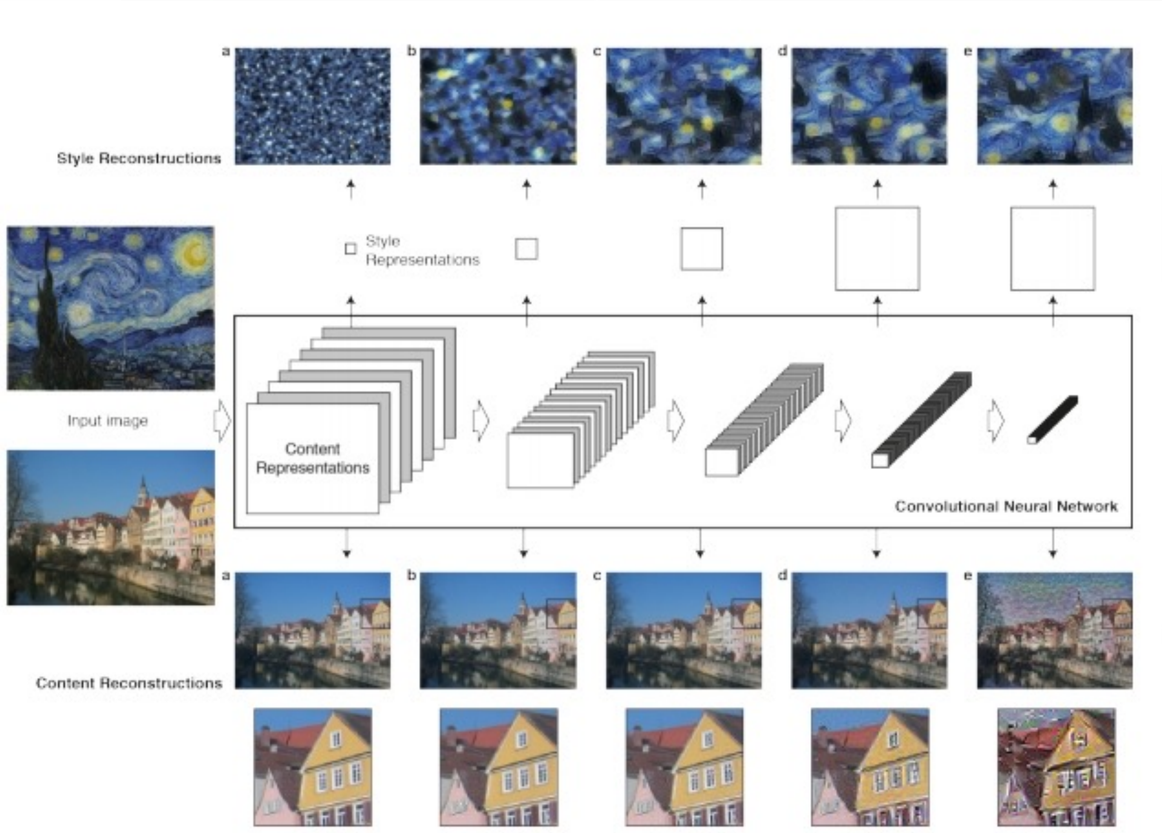
- Tensorflow框架
- 图像风格迁移原理
- 风格化模型
- 损失函数
- 优化方式
- 参数调试

案例讲解-风格迁移原理

基于图像的风格迁移原理(neural style transfer)



什么是图像风格迁移

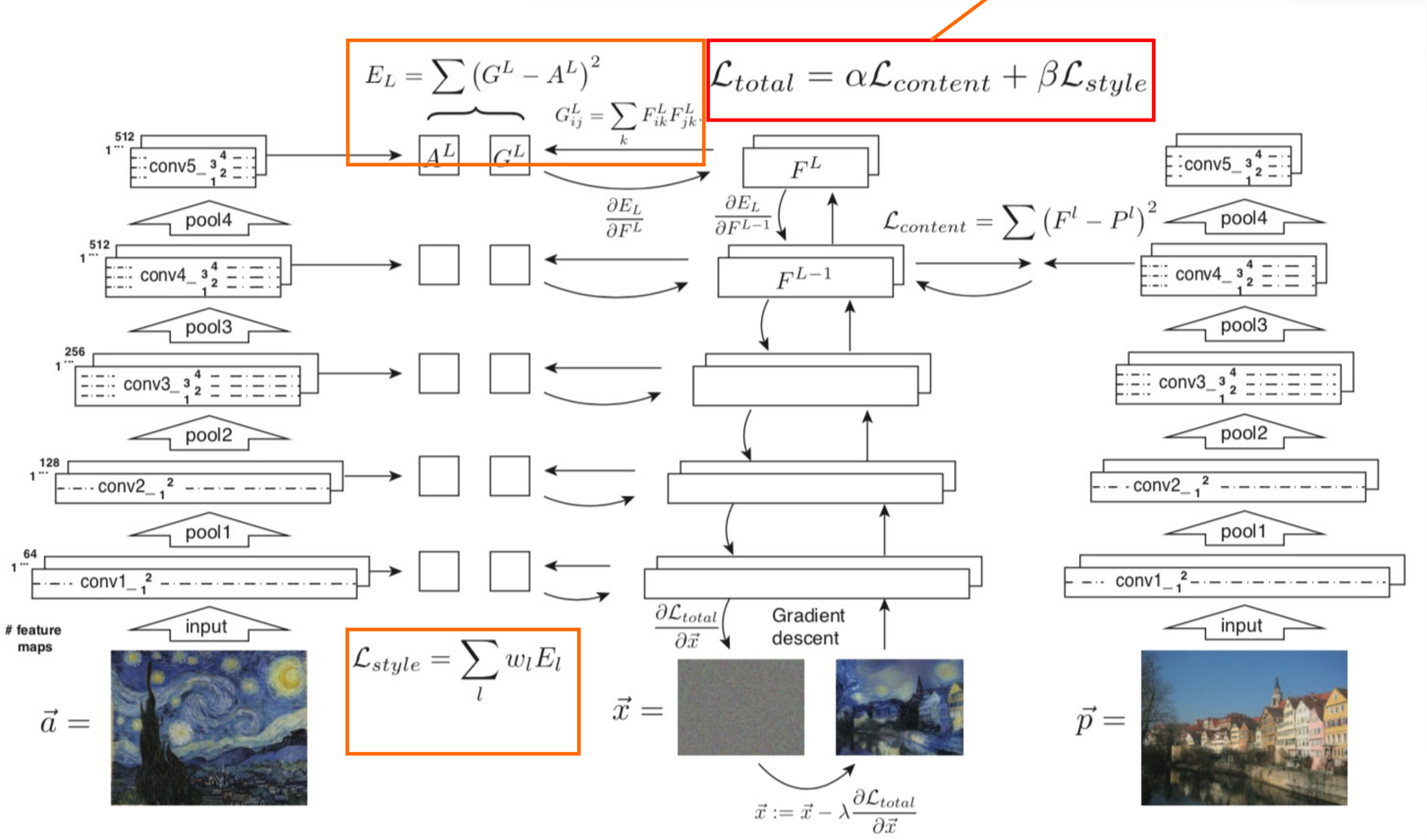


基于图像的风格迁移基本原理

案例讲解-风格迁移原理

图像风格重建+内容重建

内容损失：L2距离



风格表示：多尺度的Gram(格拉姆)矩阵

$$\Delta(\alpha_1, \alpha_2, \cdots, \alpha_k) = \begin{pmatrix} (\alpha_1, \alpha_1) & \cdots & (\alpha_1, \alpha_k) \\ \vdots & \ddots & \vdots \\ (\alpha_k, \alpha_1) & \cdots & (\alpha_k, \alpha_k) \end{pmatrix}$$

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

风格损失：风格特征距离

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2$$

案例讲解-模型解读

TIANCHI天池

VGG19预训练模型

```
## 模型定义函数
def build_model(input_img):
  net = {}
  _, h, w, d     = input_img.shape ##获得输入图像尺寸
  vgg_rawnet     = scipy.io.loadmat(args.model_weights) ##载入预训练模型
  vgg_layers     = vgg_rawnet['layers'][0]
  net['input']   = tf.Variable(np.zeros((1, h, w, d), dtype=np.float32))
  net['conv1_1'] = conv_layer('conv1_1', net['input'], W=get_weights(vgg_layers, 0))
  net['relu1_1'] = relu_layer('relu1_1', net['conv1_1'], b=get_bias(vgg_layers, 0))
  net['conv1_2'] = conv_layer('conv1_2', net['relu1_1'], W=get_weights(vgg_layers, 2))
  net['relu1_2'] = relu_layer('relu1_2', net['conv1_2'], b=get_bias(vgg_layers, 2))
  net['pool1']   = pool_layer('pool1', net['relu1_2'])
  net['conv2_1'] = conv_layer('conv2_1', net['pool1'], W=get_weights(vgg_layers, 5))
  net['relu2_1'] = relu_layer('relu2_1', net['conv2_1'], b=get_bias(vgg_layers, 5)
  net['conv2_2'] = conv_layer('conv2_2', net['relu2_1'], W=get_weights(vgg_layers, 7))
  net['relu2_2'] = relu_layer('relu2_2', net['conv2_2'], b=get_bias(vgg_layers, 7))
  net['pool2']   = pool_layer('pool2', net['relu2_2'])
  net['conv3_1'] = conv_layer('conv3_1', net['pool2'], W=get_weights(vgg_layers, 10))
  net['relu3_1'] = relu_layer('relu3_1', net['conv3_1'], b=get_bias(vgg_layers, 10))
  net['conv3_2'] = conv_layer('conv3_2', net['relu3_1'], W=get_weights(vgg_layers, 12))
  net['relu3_2'] = relu_layer('relu3_2', net['conv3_2'], b=get_bias(vgg_layers, 12))
  net['conv3_3'] = conv_layer('conv3_3', net['relu3_2'], W=get_weights(vgg_layers, 14))
  net['relu3_3'] = relu_layer('relu3_3', net['conv3_3'], b=get_bias(vgg_layers, 14))
```

```
  net['conv3_4'] = conv_layer('conv3_4', net['relu3_3'], W=get_weights(vgg_layers, 16))
  net['relu3_4'] = relu_layer('relu3_4', net['conv3_4'], b=get_bias(vgg_layers, 16))
  net['pool3']   = pool_layer('pool3', net['relu3_4'])
  net['conv4_1'] = conv_layer('conv4_1', net['pool3'], W=get_weights(vgg_layers, 19))
  net['relu4_1'] = relu_layer('relu4_1', net['conv4_1'], b=get_bias(vgg_layers, 19))
  net['conv4_2'] = conv_layer('conv4_2', net['relu4_1'], W=get_weights(vgg_layers, 21))
  net['relu4_2'] = relu_layer('relu4_2', net['conv4_2'], b=get_bias(vgg_layers, 21))
  net['conv4_3'] = conv_layer('conv4_3', net['relu4_2'], W=get_weights(vgg_layers, 23))
  net['relu4_3'] = relu_layer('relu4_3', net['conv4_3'], b=get_bias(vgg_layers, 23))
  net['conv4_4'] = conv_layer('conv4_4', net['relu4_3'], W=get_weights(vgg_layers, 25))
  net['relu4_4'] = relu_layer('relu4_4', net['conv4_4'], b=get_bias(vgg_layers, 25))
  net['pool4']   = pool_layer('pool4', net['relu4_4'])
  net['conv5_1'] = conv_layer('conv5_1', net['pool4'], W=get_weights(vgg_layers, 28))
  net['relu5_1'] = relu_layer('relu5_1', net['conv5_1'], b=get_bias(vgg_layers, 28))
  net['conv5_2'] = conv_layer('conv5_2', net['relu5_1'], W=get_weights(vgg_layers, 30))
  net['relu5_2'] = relu_layer('relu5_2', net['conv5_2'], b=get_bias(vgg_layers, 30))
  net['conv5_3'] = conv_layer('conv5_3', net['relu5_2'], W=get_weights(vgg_layers, 32))
  net['relu5_3'] = relu_layer('relu5_3', net['conv5_3'], b=get_bias(vgg_layers, 32))
  net['conv5_4'] = conv_layer('conv5_4', net['relu5_3'], W=get_weights(vgg_layers, 34))
  net['relu5_4'] = relu_layer('relu5_4', net['conv5_4'], b=get_bias(vgg_layers, 34))
  net['pool5']   = pool_layer('pool5', net['relu5_4'])
  return net
```

案例讲解-模型解读

VGG19预训练模型：基础函数

```
## 卷积层函数
def conv_layer(layer_name, layer_input, W):
  conv = tf.nn.conv2d(layer_input, W, strides=[1, 1, 1, 1], padding='SAME')
  return conv
```

```
## 激活层函数
def relu_layer(layer_name, layer_input, b):
  relu = tf.nn.relu(layer_input + b)
  return relu
```

```
## 池化层函数
def pool_layer(layer_name, layer_input):
  if args.pooling_type == 'avg':
    pool = tf.nn.avg_pool(layer_input, ksize=[1, 2, 2, 1],
      strides=[1, 2, 2, 1], padding='SAME')
  elif args.pooling_type == 'max':
    pool = tf.nn.max_pool(layer_input, ksize=[1, 2, 2, 1],
      strides=[1, 2, 2, 1], padding='SAME')
  return pool
```

```
## 权重获取函数
def get_weights(vgg_layers, i):
  weights = vgg_layers[i][0][0][2][0][0]
  W = tf.constant(weights)
  return W

## 偏置获取函数
def get_bias(vgg_layers, i):
  bias = vgg_layers[i][0][0][2][0][1]
  b = tf.constant(np.reshape(bias, (bias.size)))
  return b
```

案例讲解-损失定义

内容损失

```
## 单个网络层的内容损失
def content_layer_loss(p, x):
  _, h, w, d = p.get_shape() #获得参考图的尺寸，这里的x和p大小相等
  M = h.value * w.value ##图像大小
  N = d.value ##图像维度
  ## 计算不同的归一化方法的权重因子
  if args.content_loss_function   == 1:
    K = 1. / (2. * N**0.5 * M**0.5)
  elif args.content_loss_function == 2:
    K = 1. / (N * M)
  elif args.content_loss_function == 3:
    K = 1. / 2.
  loss = K * tf.reduce_sum(tf.pow((x - p), 2)) ##计算L2损失
  return loss

## 总的内容损失函数
def sum_content_losses(sess, net, content_img):
  sess.run(net['input'].assign(content_img))
  content_loss = 0.
  for layer, weight in zip(args.content_layers, args.content_layer_weights):
    p = sess.run(net[layer])
    x = net[layer]
    p = tf.convert_to_tensor(p)
    content_loss += content_layer_loss(p, x) * weight
  content_loss /= float(len(args.content_layers))
  return content_loss
```

args.content_layers配置了网络层，
args.content_layer_weights配置了该层的权重，
只使用VGG19中的conv4_2层

案例讲解-损失定义

TIANCHI天池

风格损失

```
## Gram矩阵计算函数
## tf.transpose完成转置，tf.matmul完成乘法计算
def gram_matrix(x, area, depth):
  F = tf.reshape(x, (area, depth))
  G = tf.matmul(tf.transpose(F), F)
  return G

## 单个网络层的风格损失
def style_layer_loss(a, x):
  _, h, w, d = a.get_shape()
  M = h.value * w.value
  N = d.value
  A = gram_matrix(a, M, N)
  G = gram_matrix(x, M, N)
  loss = (1./(4 * N**2 * M**2)) * tf.reduce_sum(tf.pow((G - A), 2))
  return loss
```

```
## 多个风格图的损失计算
def sum_style_losses(sess, net, style_imgs):
  total_style_loss = 0.
  weights = args.style_imgs_weights ##获得每一层风格层的权重
  for img, img_weight in zip(style_imgs, weights): ##遍历各个风格图
    sess.run(net['input'].assign(img))
    style_loss = 0.

## 计算每一个风格图的风格损失
    for layer, weight in zip(args.style_layers, args.style_layer_weights):
      a = sess.run(net[layer])
      x = net[layer]
      a = tf.convert_to_tensor(a)
      style_loss += style_layer_loss(a, x) * weight
      style_loss /= float(len(args.style_layers))

      #使用每一张风格图的权重进行加权获得最后的损失
      total_style_loss += (style_loss * img_weight)

  total_style_loss /= float(len(style_imgs)
  return total_style_loss
```

使用VGG19的5个网络层的输出，分别是'relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', 'relu5_1'

案例讲解-模型训练

TIANCHI天池

模型训练

```
## 模型训练函数
def stylize(content_img, style_imgs, init_img, frame=None):
  with tf.device(args.device), tf.Session() as sess:
    # 定义模型
    net = build_model(content_img)
    L_style = sum_style_losses(sess, net, style_imgs) # 计算风格损失
    L_content = sum_content_losses(sess, net, content_img) #计算内容损失
    L_tv = tf.image.total_variation(net['input']) #计算平滑损失
    alpha = args.content_weight #内容损失权重
    beta  = args.style_weight #风格损失权重
    theta = args.tv_weight  #噪声损失权重

    #获得总的损失
    L_total  = alpha * L_content
    L_total += beta  * L_style
    L_total += theta * L_tv

    #定义优化器，可以使用adam或者lbfgs
    optimizer = get_optimizer(L_total)
    if args.optimizer == 'adam':
      minimize_with_adam(sess, net, optimizer, init_img, L_total)
    elif args.optimizer == 'lbfgs':
      minimize_with_lbfgs(sess, net, optimizer, init_img)

    output_img = sess.run(net['input']) ##得到输出图
```

案例讲解-优化目标与方法

TIANCHI天池

优化方法：Adam算法

```
## 优化函数
def get_optimizer(loss):
  if args.optimizer == 'lbfgs': ##lbfgs优化器
    optimizer = tf.contrib.opt.ScipyOptimizerInterface(
      loss, method='L-BFGS-B',
      options={'maxiter': args.max_iterations,
               'disp': print_iterations})
  elif args.optimizer == 'adam': #adam优化器
    optimizer = tf.train.AdamOptimizer(args.learning_rate)
  return optimizer
```

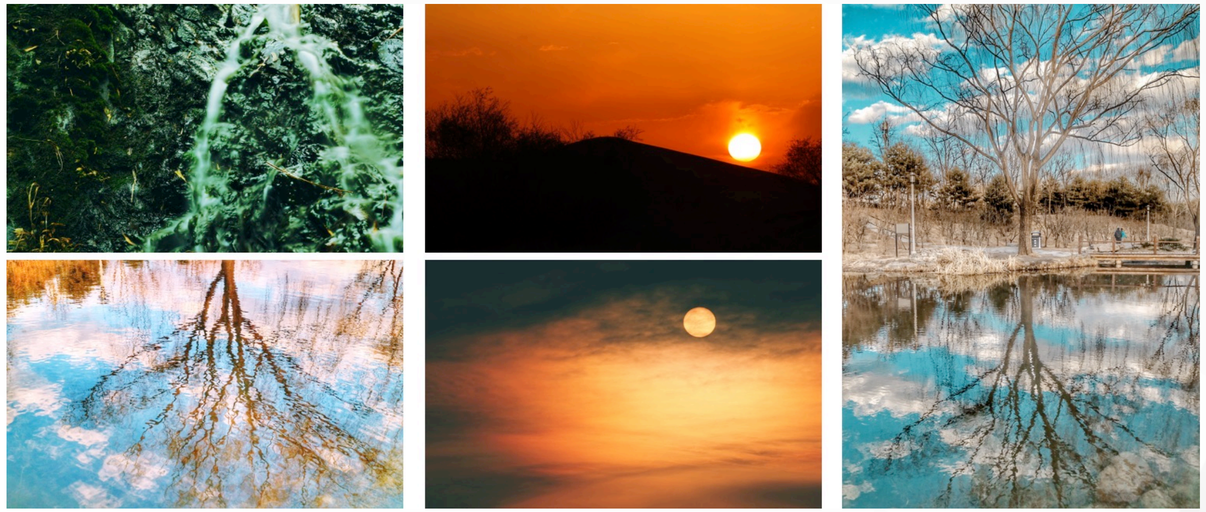
```
## Adam优化器
def minimize_with_adam(sess, net, optimizer, init_img, loss):
  train_op = optimizer.minimize(loss)
  init_op = tf.global_variables_initializer()
  sess.run(init_op)
  sess.run(net['input'].assign(init_img))
  iterations = 0
  while (iterations < args.max_iterations):
    sess.run(train_op)
    if iterations % args.print_iterations == 0 and args.verbose:
      curr_loss = loss.eval()
      print("At iterate {}\tf=  {}".format(iterations, curr_loss))
iterations += 1
```

案例讲解-训练迭代

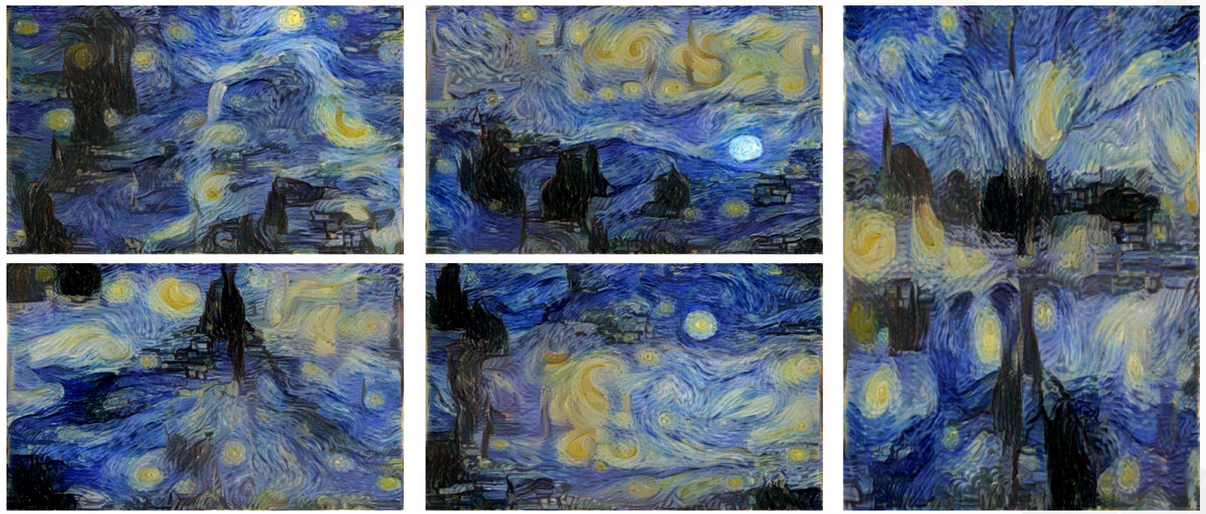
TIANCHI天池

准备风格图与若干内容图

风格图



内容图



结果图

直播相关资料获取及回放查看地址： https://tianchi.aliyun.com/specials/promotion/activity/bookclub

案例讲解-更多测试

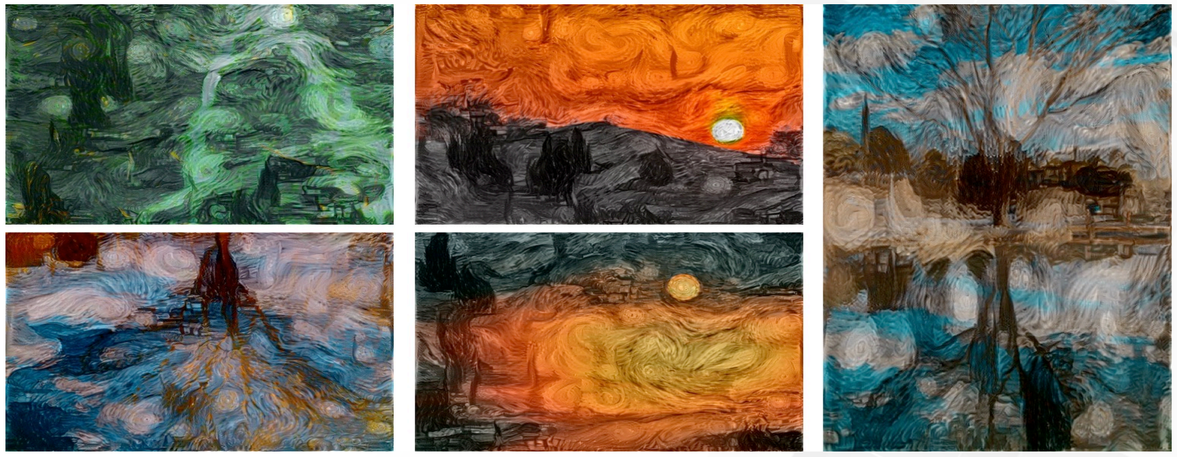
TIANCHI天池

颜色保持

```
def convert_to_original_colors(content_img, stylized_img):
  ##后处理操作，即添加均值，将RGB图像转换为BGR图像
  content_img  = postprocess(content_img)
  stylized_img = postprocess(stylized_img)

  ## 颜色空间，可选包括yuv，ycrcb，luv，lab
  if args.color_convert_type == 'yuv':
    cvt_type = cv2.COLOR_BGR2YUV
    inv_cvt_type = cv2.COLOR_YUV2BGR
  elif args.color_convert_type == 'ycrcb':
    cvt_type = cv2.COLOR_BGR2YCR_CB
    inv_cvt_type = cv2.COLOR_YCR_CB2BGR
  elif args.color_convert_type == 'luv':
    cvt_type = cv2.COLOR_BGR2LUV
    inv_cvt_type = cv2.COLOR_LUV2BGR
  elif args.color_convert_type == 'lab':
    cvt_type = cv2.COLOR_BGR2LAB
  inv_cvt_type = cv2.COLOR_LAB2BGR

  ## 将内容图和结果图转换到对应的颜色空间
  content_cvt = cv2.cvtColor(content_img, cvt_type)
  stylized_cvt = cv2.cvtColor(stylized_img, cvt_type)

  ## 取结果图的亮度通道和内容图的两个颜色通道组合成新的图
  c1, _, _ = cv2.split(stylized_cvt)
  _, c2, c3 = cv2.split(content_cvt)
  merged = cv2.merge((c1, c2, c3))
  dst = cv2.cvtColor(merged, inv_cvt_type).astype(np.float32)

  ## 预处理
  dst = preprocess(dst)
  return dst
```

```
def postprocess(img):
  imgpost = np.copy(img)
  imgpost += np.array([123.68, 116.779, 103.939]).reshape((1,1,1,3)) ##添加均值
  imgpost = imgpost[0]   # 将四维张量(1, h, w, d)转换为三维张量(h, w, d)
  imgpost = np.clip(imgpost, 0, 255).astype('uint8') ##浮点型图像转换为uint8型
  imgpost = imgpost[...,::-1] #RGB图像转换为BGR图像
  return imgpost
```

```
def preprocess(img):
  imgpre = np.copy(img)
  imgpre = imgpre[...,::-1] #BGR图像转换为RGB图像
  imgpre = imgpre[np.newaxis,:,:,:] # 将三维张量(h, w, d)转换为四维张量(1, h, w, d)
  imgpre -= np.array([123.68, 116.779, 103.939]).reshape((1,1,1,3)) ##减去均值
  return imgpre
```



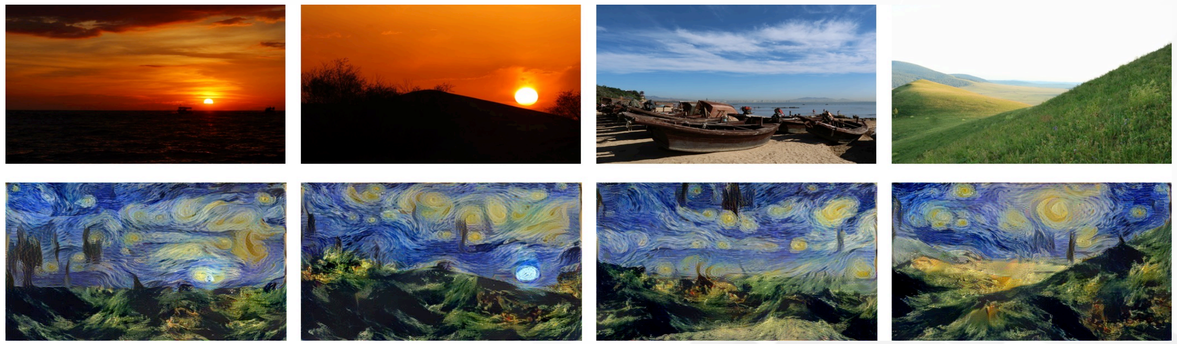
案例讲解-更多测试

TIANCHI天池

语义信息保持

```
## 带掩膜的风格损失计算
def sum_masked_style_losses(sess, net, style_imgs):
  total_style_loss = 0.
  weights = args.style_imgs_weights
  masks = args.style_mask_imgs ##获得每一种风格的掩膜
  for img, img_weight, img_mask in zip(style_imgs, weights, masks):
    sess.run(net['input'].assign(img))
    style_loss = 0.
    for layer, weight in zip(args.style_layers, args.style_layer_weights):
      a = sess.run(net[layer])
      x = net[layer]
      a = tf.convert_to_tensor(a)
      a, x = mask_style_layer(a, x, img_mask) ##根据掩膜mask计算损失
      style_loss += style_layer_loss(a, x) * weight
    style_loss /= float(len(args.style_layers))
    total_style_loss += (style_loss * img_weight)

  total_style_loss /= float(len(style_imgs))
  return total_style_loss
```

```
## 掩膜风格层具体实现
def mask_style_layer(a, x, mask_img):
  _, h, w, d = a.get_shape()
  mask = get_mask_image(mask_img, w.value, h.value)
  mask = tf.convert_to_tensor(mask) ##获得掩膜tensor
  tensors = []
  for _ in range(d.value):
    tensors.append(mask)
  mask = tf.stack(tensors, axis=2)
  mask = tf.stack(mask, axis=0)
  mask = tf.expand_dims(mask, 0)
  ##结果与掩膜相乘，掩膜为0的地方结果也为0
  a = tf.multiply(a, mask)
  x = tf.multiply(x, mask)
  return a, x
```



实战演示、互动交流

大家可以使用手机扫左侧海报二维码，或者**电脑**访问下方地址进入天池读书会页面，点击今天读书会中的 **实践代码** 和我一起进行项目实践学习，天池为大家准备好了代码和运行环境，非常方便。

https://tianchi.aliyun.com/specials/promotion/activity/bookclub

言有三 本书作者、有三AI创始人

直播主题 《深度学习之摄影图像处理 核心算法与案例精粹》
直播时间 2021年5月25日 20:00
学习资料 深度学习训练营
实践项目 图像风格化实战

提问 | 学习训练营 | 购买地址 | PPT下载 | 实践代码 | 预约直播

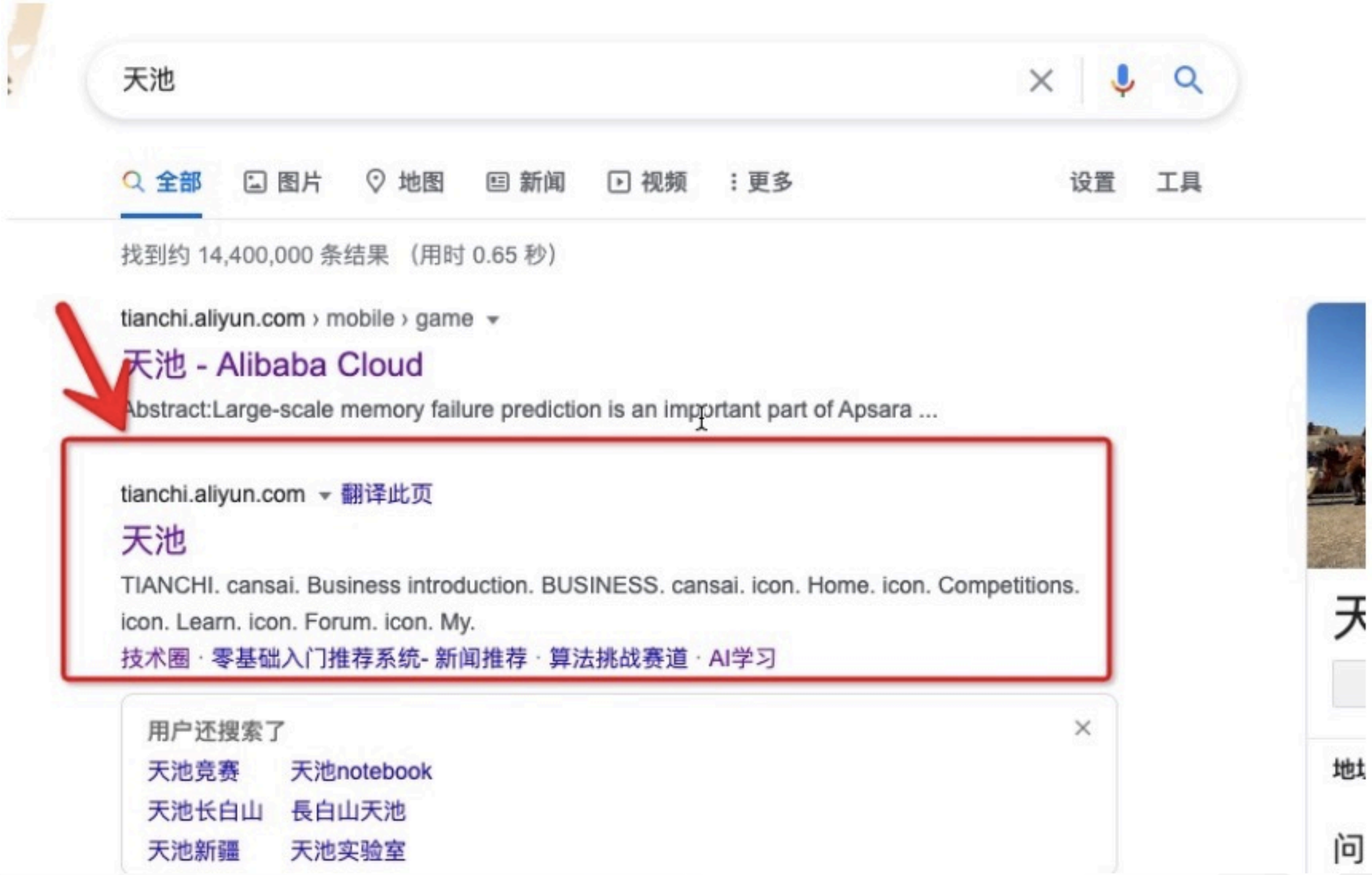
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

TIANCHI天池

Q&A

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

1）首先需要进入天池官网，大家打开浏览器，搜索 天池 ，找到 tianchi.aliyun.com即可访问进入天池官网；



2）在天池官网，将鼠标移到 天池学习 ，即可出现下拉列表，点击 天池读书会 ，即可进入天池读书会的页面。

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

3）在天池读书会页面，你可以对对应的读书会图书进行提问，优秀的提问还有机会获得赠书，还可以点击配套的训练营或者课程资源进入学习，还有点击实践代码获取读书会的项目实践的代码，跟着我一起进行项目实践和代码学习，同时还有很多其他的读书会，大家也可以观看举办过的读书会的回放，或者预约还没开始的读书会。

言有三 本书作者、有三AI创始人

直播主题 《深度学习之摄影图像处理 核心算法与案例精粹》

直播时间 2021年5月25日 20:00

学习资料 深度学习训练营

实践项目 图像风格化实战

提问 | 学习训练营 | 购买地址 | PPT下载 | 实践代码 | 预约直播

直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

天池读书会

TIANCHI天池

异步社区
人民邮电出版社

《深度学习之摄影图像处理 核心算法与案例精粹》

从摄影作品的美学评价算法，自动构图算法，过渡到对

摄影作品的常用编辑技术，内容由浅入深，覆盖了大量应用场景

直播嘉宾：言有三

直播时间：5月25日 20:00

扫码领取读书会配套学习资源