阿里云天池牛年读书会

人脸图像算法与典型实践

分享嘉宾：言有三
个人简介(深度学习算法专家)

抽奖提示

言有三 有三AI创始人、本书作者

直播主题 《深度学习之人脸图像处理》

直播时间 2021年2月27日 20:00

学习资料 课程《深度学习原理与实践》

实践项目 深度学习训练营

提问 | 课程 | 训练营 | 实践代码 | 预约直播

阿里云

牛年读书会提问问卷

本问卷主要用于收集大家对直播分享的图书的相关问题，作者将选取一些有意义和广泛的问题为大家解答。
我们会定期将大家问题反馈给分享嘉宾，并将问题和作者解答分享到天池技术圈：点击我直达技术圈
往期读书会视频回放查看地址：点击我直接查看

* 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？(可多选)

- ☐ 《Python数据分析挖掘与可视化》
- ☐ 《机器学习的数学》
- ☐ 《机器学习原理》
- ☐ 《阿里云天池赛题解析机器学习篇》
- ☐ 《动手学深度学习》
- ☑ 《深度学习之人脸图像处理》
- ☐ 《数据分析通识》
- ☐ 《？？》等你来推荐

目录

TIANCHI天池

1. 作者简介

2. 图书简介

3.项目实践

4.Q&A 答疑

作者简介

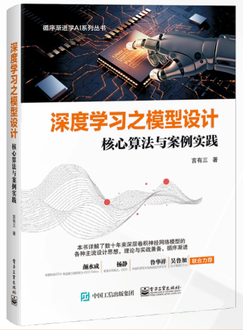
TIANCHI天池

龙鹏，笔名言有三，先后就读于华中科技大学(2008-2012)，中国科学院半导体研究所神经网络实验室(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，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)。

玩转图像识别

玩转模型设计



玩转人脸图像

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

TIANCHI天池

业界首本系统性讲述深度学习人脸图像算法的书籍，理论和实践结合

人脸图像在计算机视觉领域的应用非常广泛，几乎每一个从事计算机视觉领域相关工作的技术人员都会有所涉及。本书从人脸图像的各种应用场景出发，介绍了以深度学习模型为主的核心技术，理论知识翔实，实践案例丰富，适合从事人脸图像相关工作的技术人员阅读。

——依图科技CTO/新加坡工程院院士/IEEE Fellow 颜水成

人脸识别是计算机视觉领域比较成功的技术之一，有着大量的落地应用。其成功离不开以深度卷积神经网络为主的深度学习技术。本书全面介绍了基于深度学习的人脸识别方法及相关应用，囊括人脸检测、特征点定位、人脸识别、人脸属性估计、人脸部件分割、3D人脸重建、人脸美颜与美妆等常用技术。本书内容全面，案例丰富，实用性强，不仅适合人脸识别领域的研究人员、从业者和技术爱好者阅读，也对计算机视觉、模式识别和机器学习等领域的从业者有参考价值。

——中科院计算所研究员/智能信息处理重点实验室常务副主任 山世光

随着移动互联网的全面发展，深度学习在人脸图像领域的应用层出不穷。本书围绕深度学习技术，详细地介绍了人脸图像处理的相关知识，并配以丰富的算法实践。通过阅读本书，读者可以更加全面地了解相关的理论知识及应用，并系统地掌握人脸图像处理技术。

——美图公司MTlab负责人 许清泉

深度学习在产业界的崛起始于ImageNet。中国在人脸图像应用领域也走在了世界前沿，屡屡斩获全球大赛冠军奖项。但是业界还是缺乏系统地讲解人脸图像核心算法理论和实践的书籍。言有三在深度学习领域深耕多年，积累了丰富的人脸图像处理项目经验，他的这本书详细地介绍了人脸图像在各种应用场景中所需要的核心技术，丰富和充实了基于深度学习的人脸图像处理类图书市场。

——新智元创始人/CEO 杨静

配书资料获取方式

本书涉及的源代码文件等配书学习资料需要读者自行下载，下载方法请参阅前言中的详细说明。

扫码查看更多数字资源

投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com
客服电话：(010) 88361066 88379833 68326294

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

上架指导：计算机/人工智能
ISBN 978-7-111-66025-5
定价：119.00元

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 China Machine Press

“有三AI”大力支持，业内4位大咖力荐
作者曾供职于奇虎360人工智能研究院与陌陌深度学习实验室
凝聚作者7余年的传统图像算法与深度学习算法的研究心得和实践经验

言有三◎著

深度学习之人脸图像处理
核心算法与案例实战

Face Image Processing by Deep Learning: Core Algorithms and Practices

聚焦人脸图像处理的核心技术与典型应用场景
详解人脸图像处理的核心算法与八大经典案例

颜水成 依图科技CTO | 山世光 中科院计算所研究员 | 许清泉 美图公司MTlab负责人 | 杨静 新智元创始人/CEO | 大力推荐

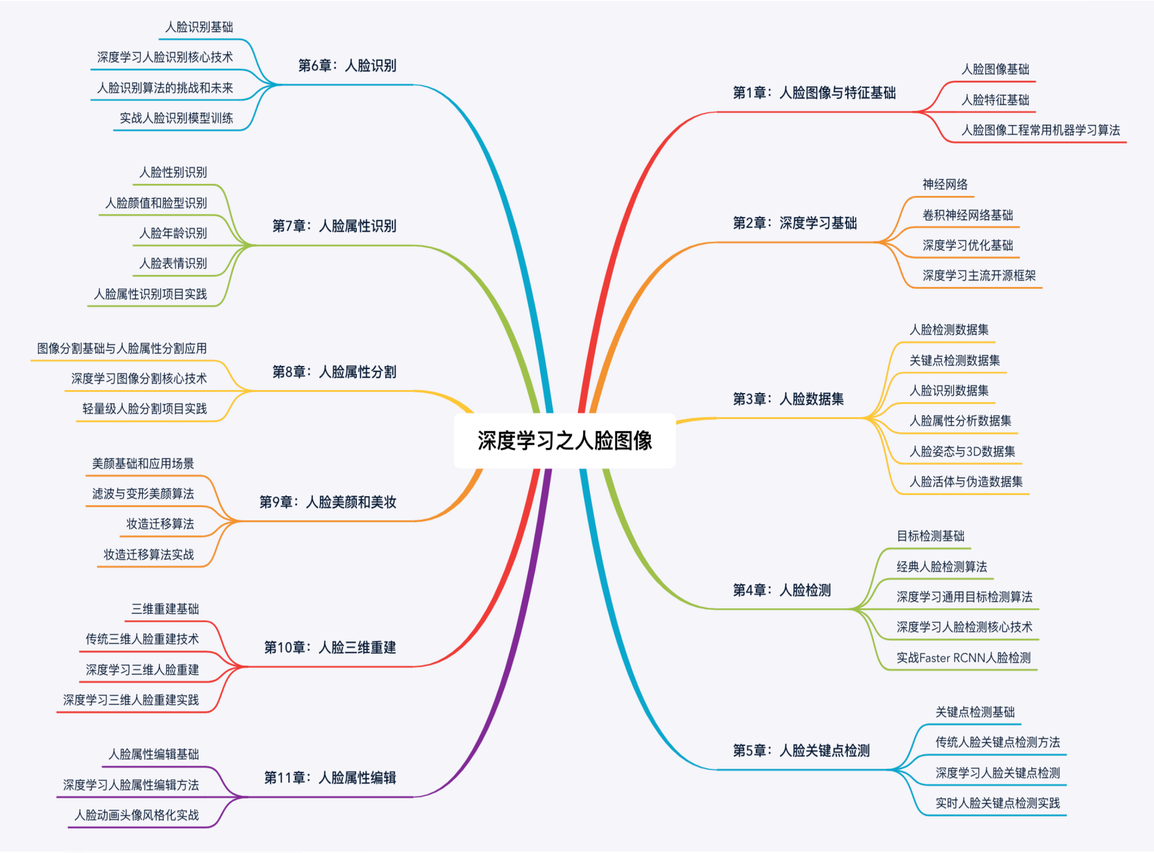
机械工业出版社 China Machine Press

第1章：人脸特征基础
第2章：深度学习基础
第3章：人脸数据集
第4章：人脸检测
第5章：人脸关键点检测
第6章：人脸识别
第7章：人脸属性识别
第8章：人脸属性分割
第9章：人脸美颜
第10章：人脸三维重建
第11章：人脸属性编辑

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

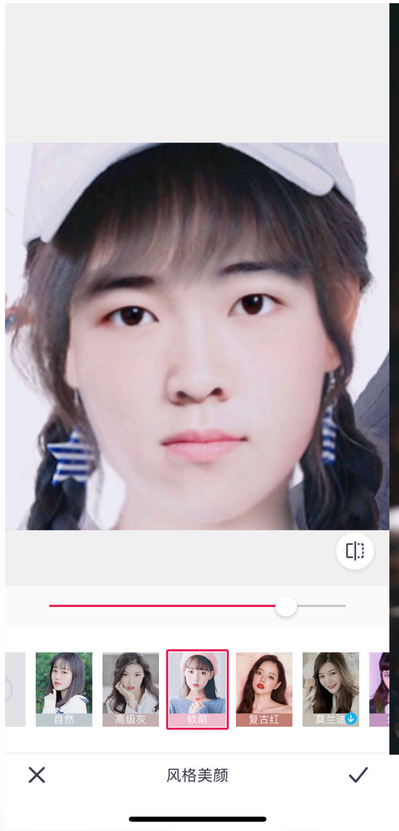


直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

应用领域

TIANCHI天池

软硬件产品



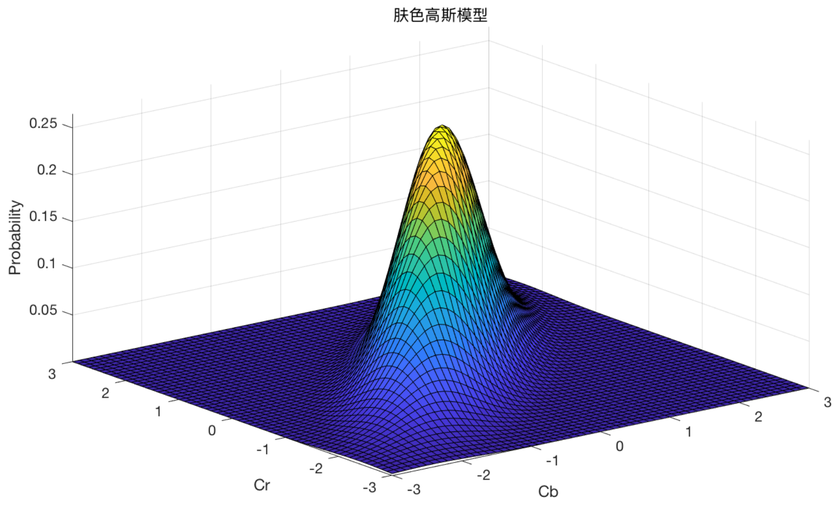
天天P图·叶问4

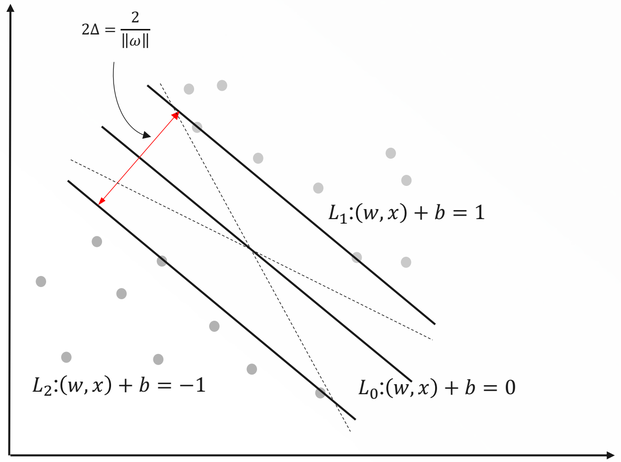
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

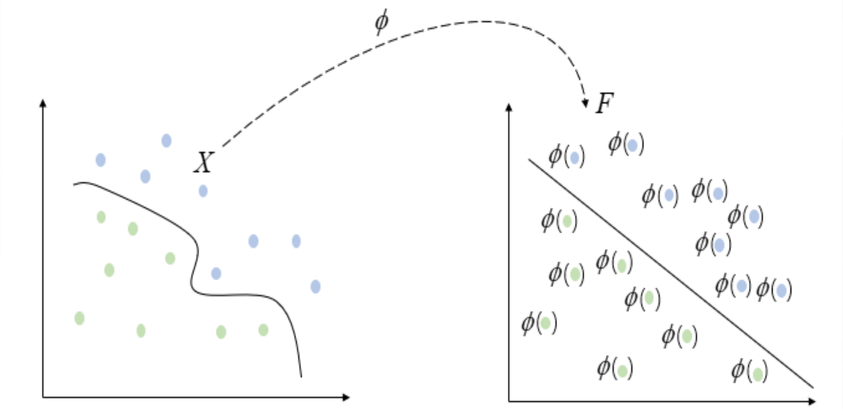
图书内容

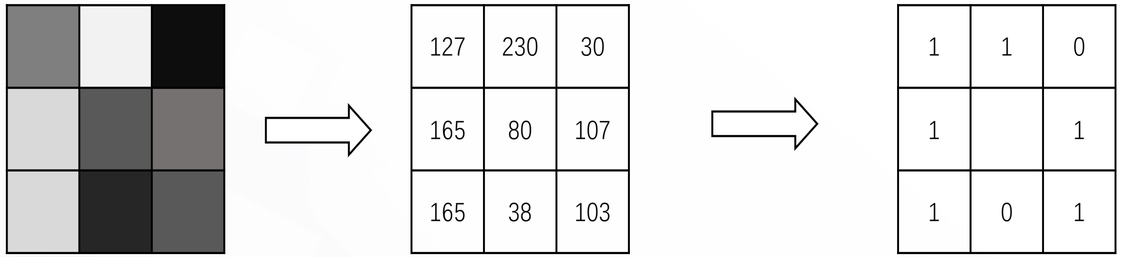
第一章：人脸特征基础

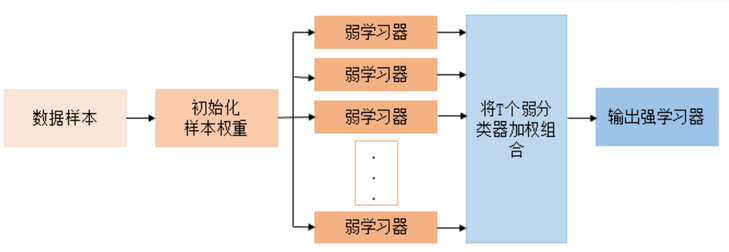
包含人脸图像基础，人脸特征基础(高斯直方图，纹理算子等)，人脸图像工程常用机器学习算法(SVM，Adaboost等)。









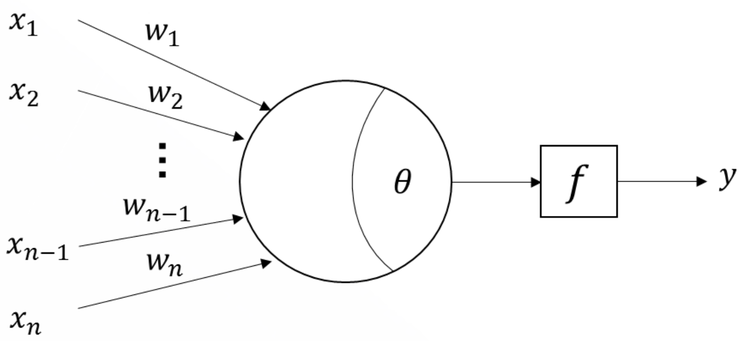


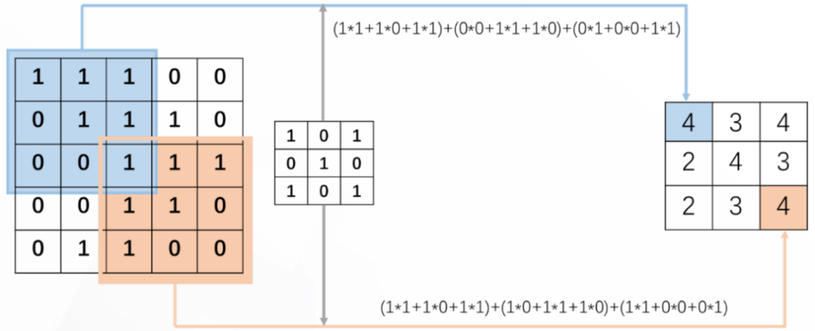
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

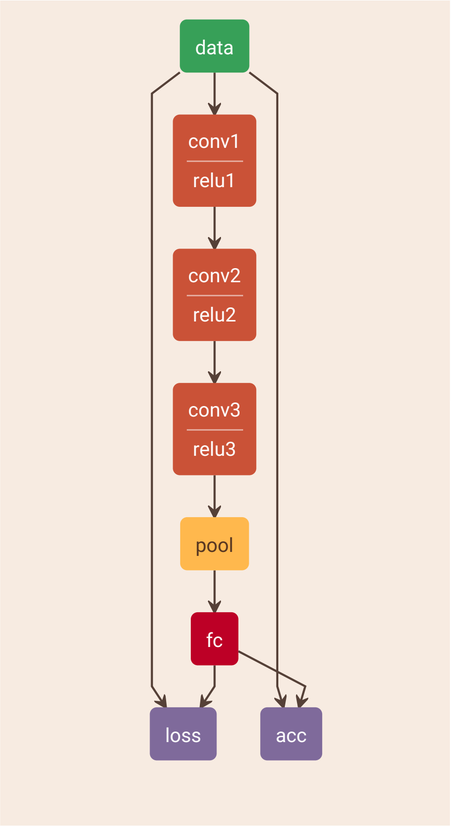
图书内容

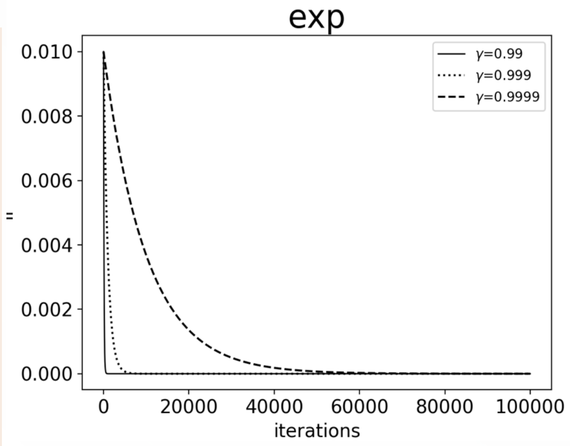
第二章：深度学习基础

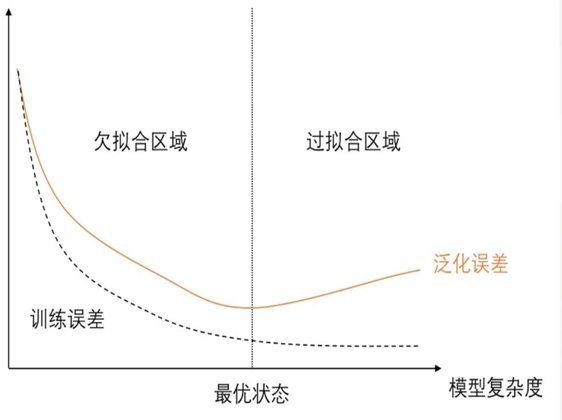
包含神经网络基础，卷积神经网络基础，深度学习优化基础(激活函数，标准化方法，泛化与正则化，优化方法等)， 深度学习开源框架简介(Tensorflow，Pytorch等)。

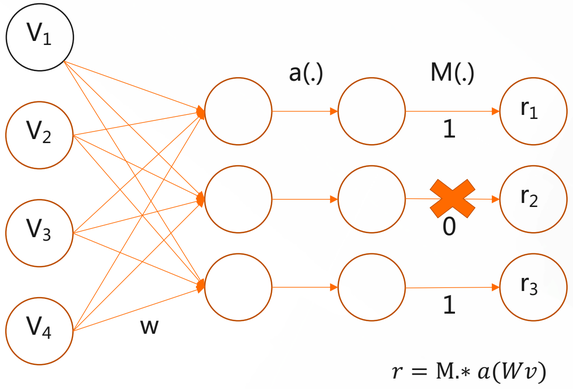


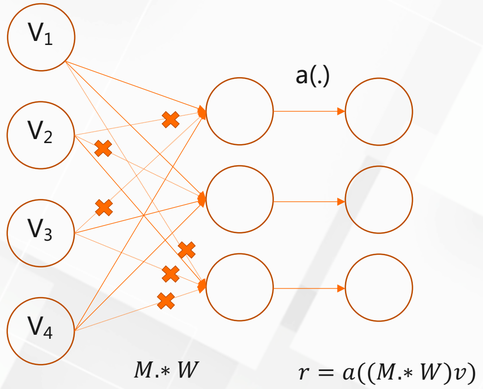










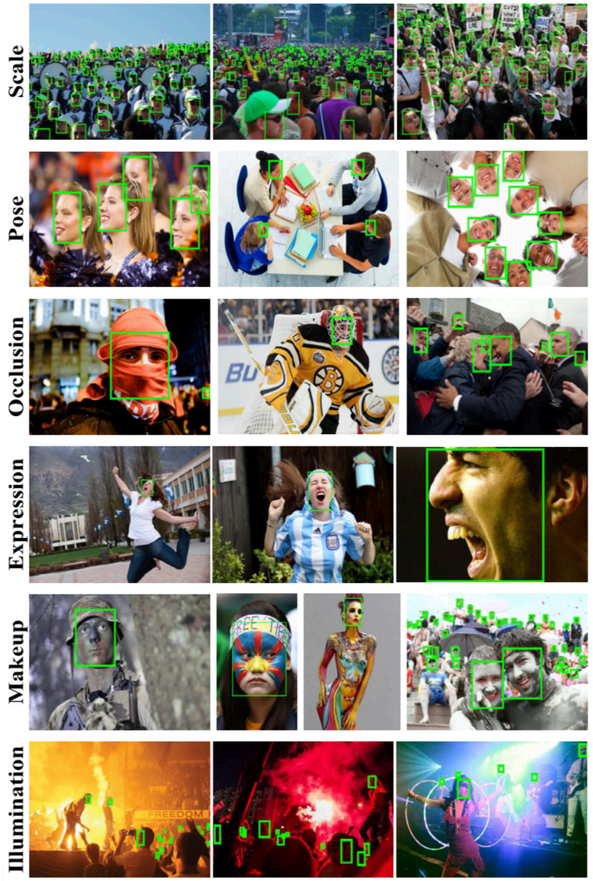


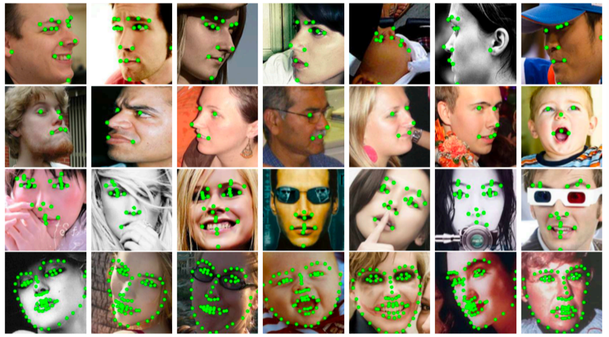
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

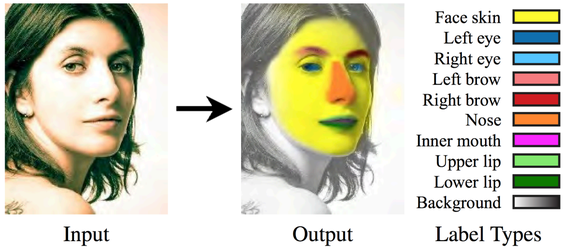
图书内容

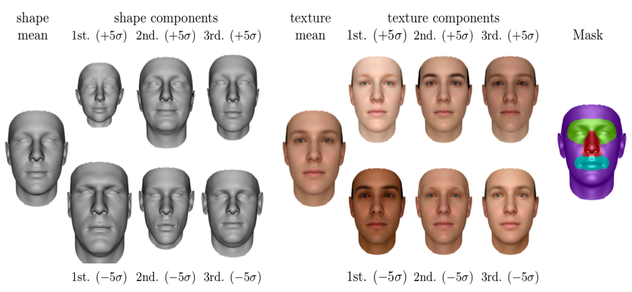
第三章：人脸数据集基础

包含人脸检测数据集，关键点检测数据集，人脸识别数据集，人脸属性分析数据集，人脸姿态与3D数据集，人脸活体与伪造数据集。







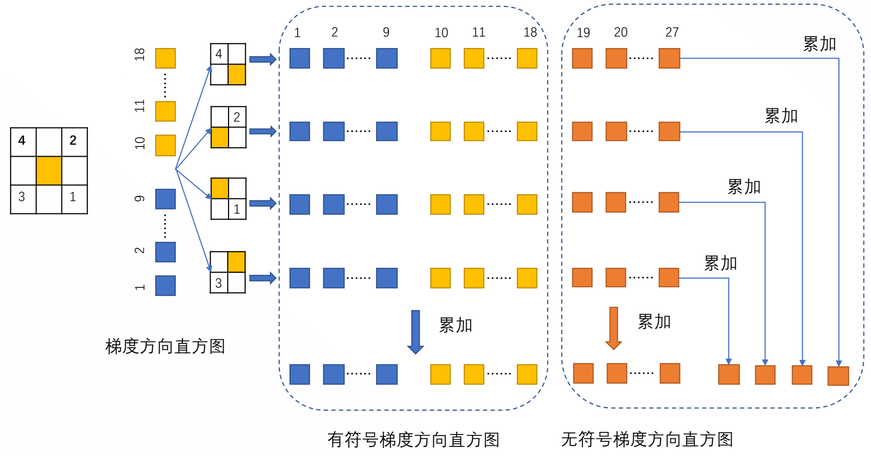


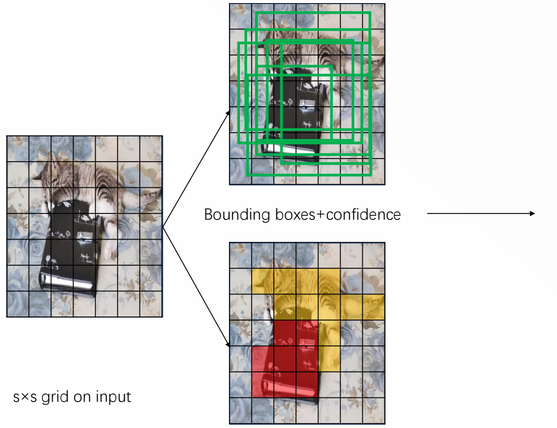
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

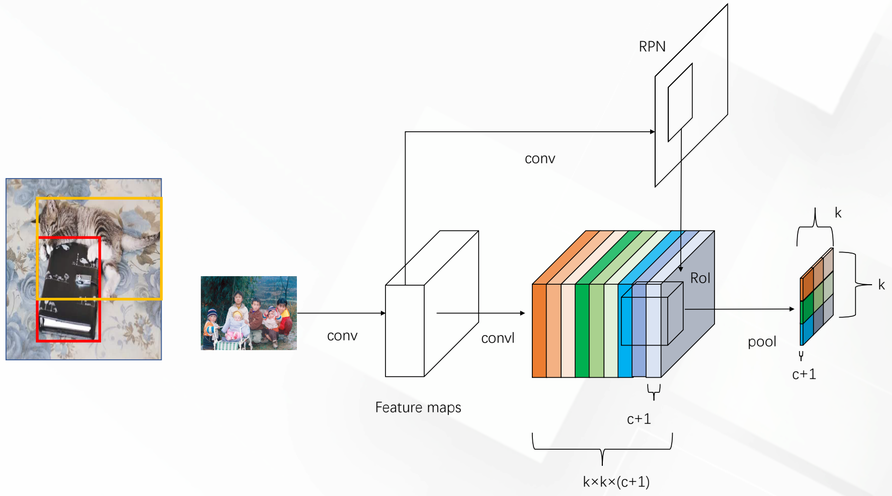
图书内容

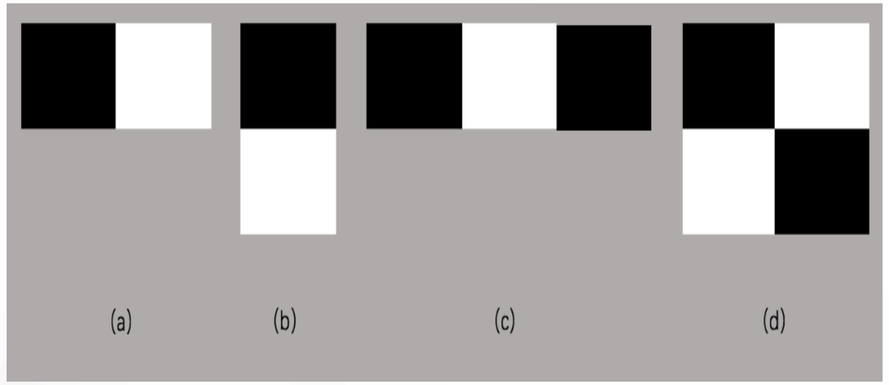
第四章：人脸检测

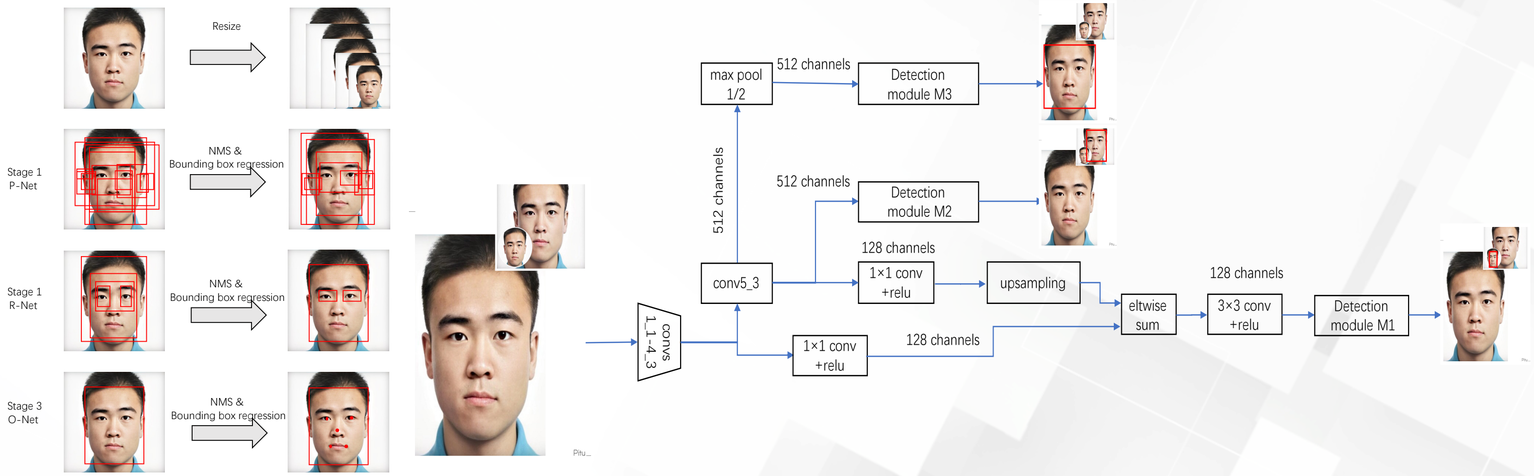
目标检测基础，经典人脸检测算法，深度学习通用目标检测算法，深度学习人脸检测核心技术，实战Faster RCNN人脸检测









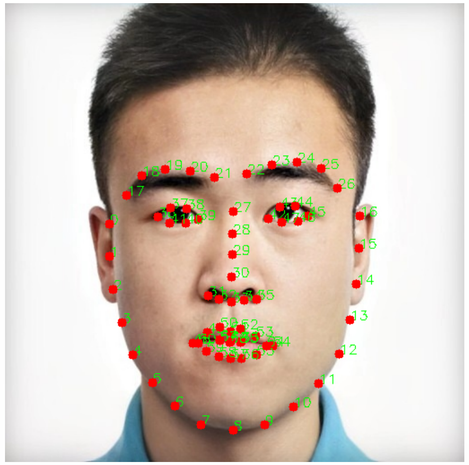


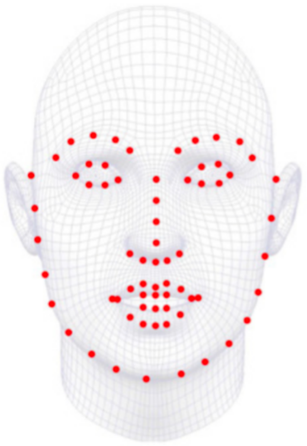
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

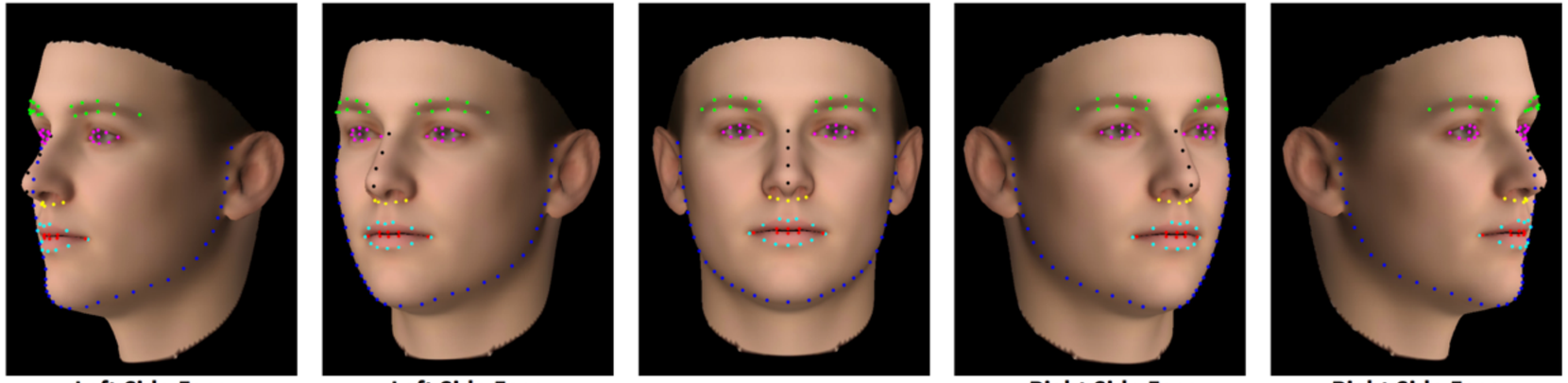
图书内容

第五章：人脸关键点检测

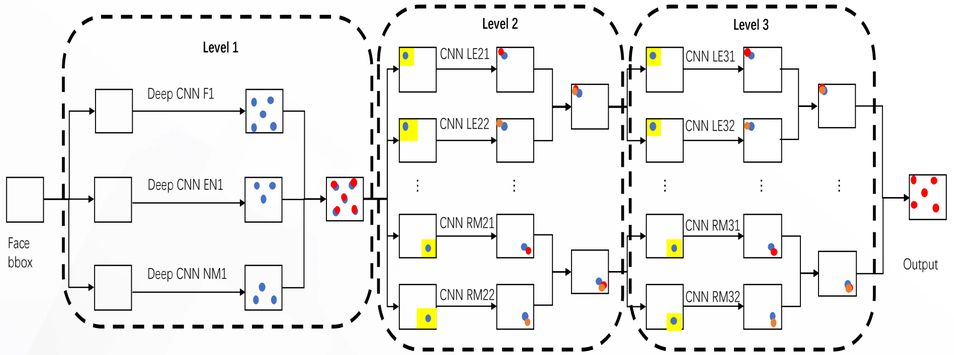
关键点检测基础，传统人脸关键点检测方法，深度学习人脸关键点检测，实时人脸关键点检测实践

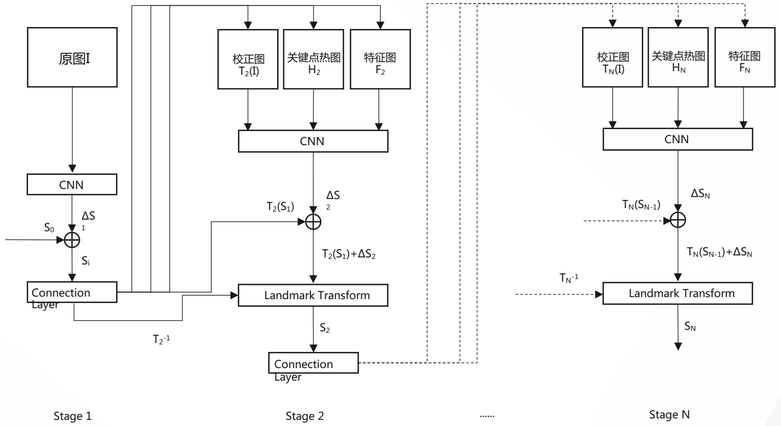


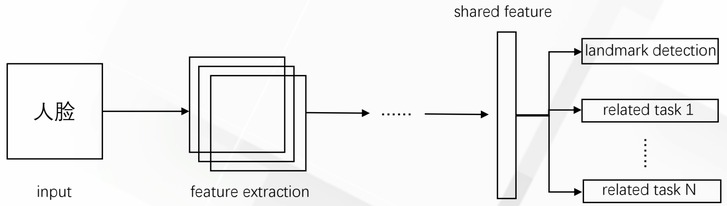




Left Side Face (Profile) | Left Side Face (Simi-Frontal) | Frontal Face | Right Side Face (Simi-Frontal) | Right Side Face (Profile)





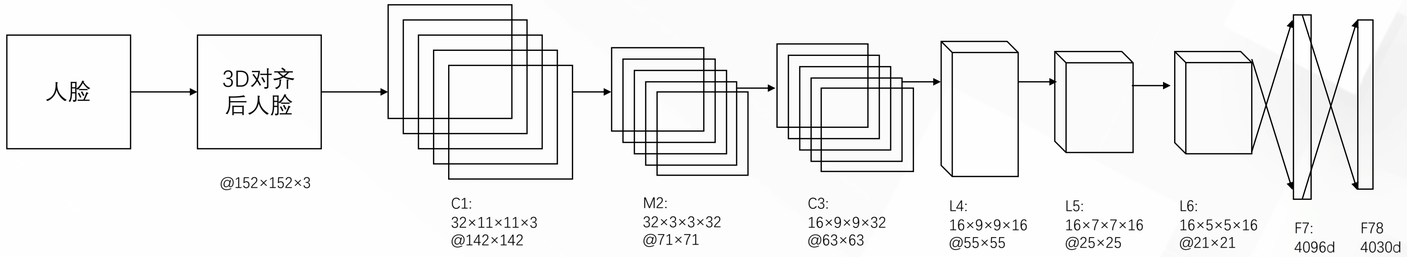


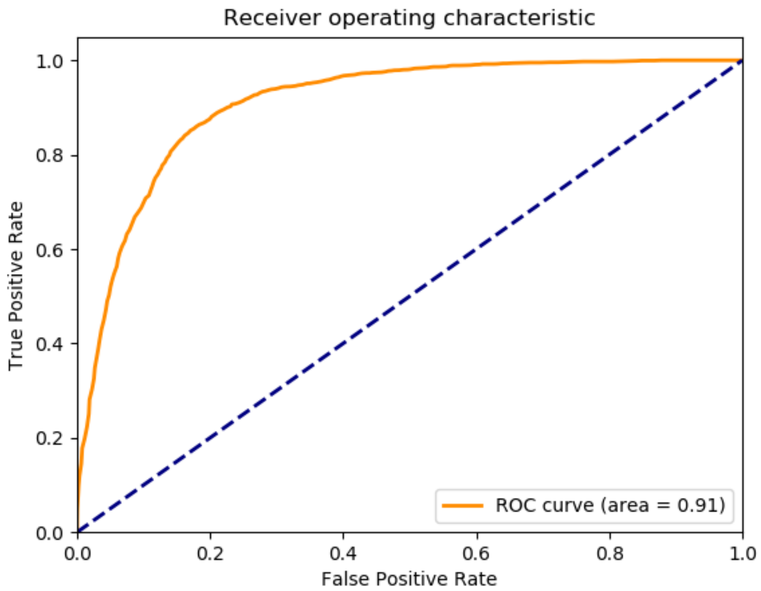
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

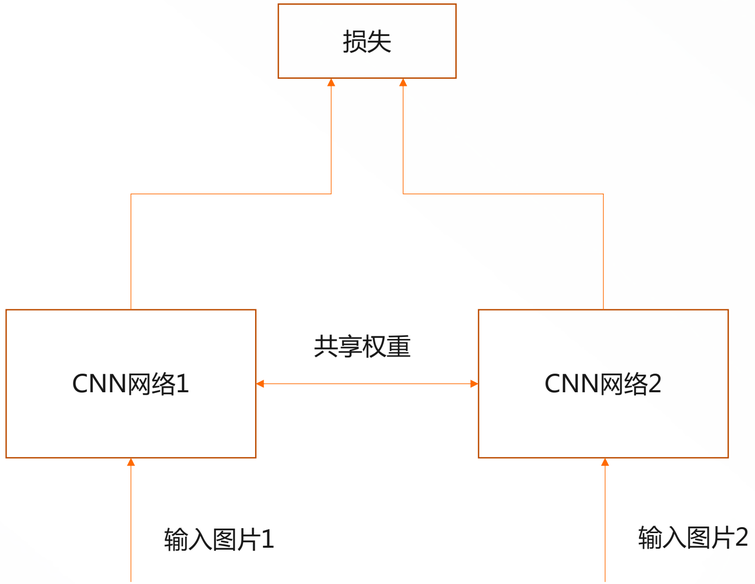
图书内容

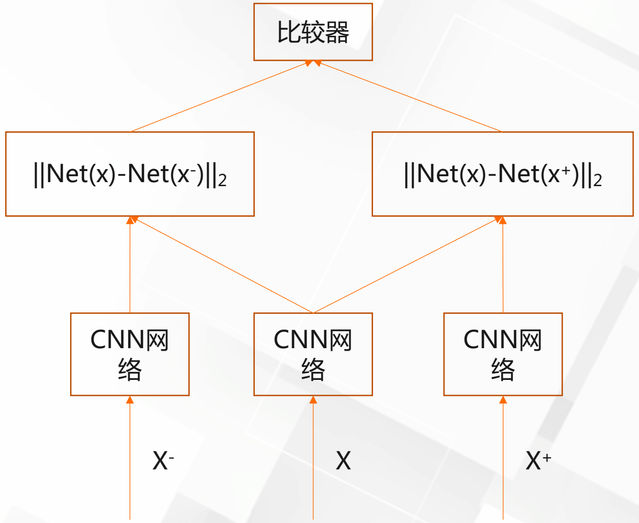
第六章：人脸识别

人脸识别基础，深度学习人脸识别核心技术，人脸识别算法的挑战和未来，实战人脸识别模型训练







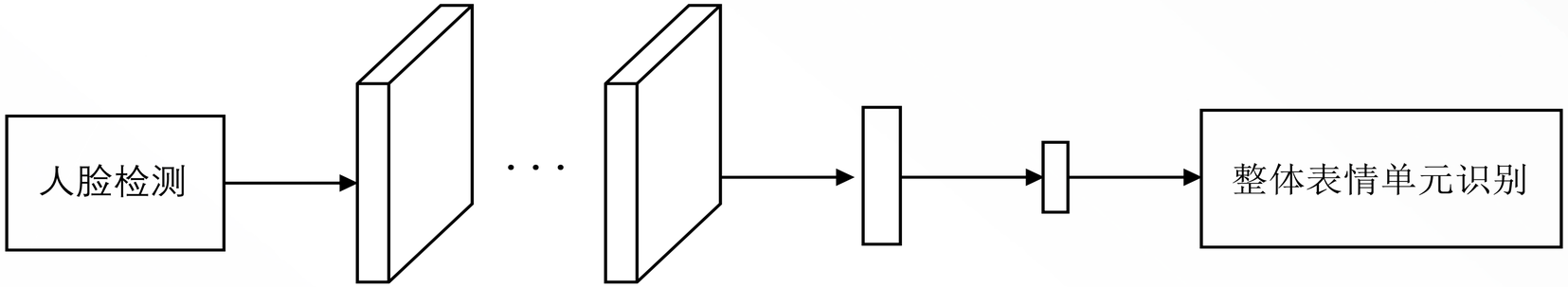


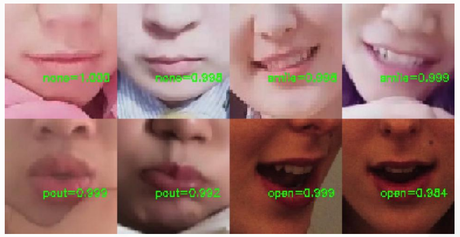
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

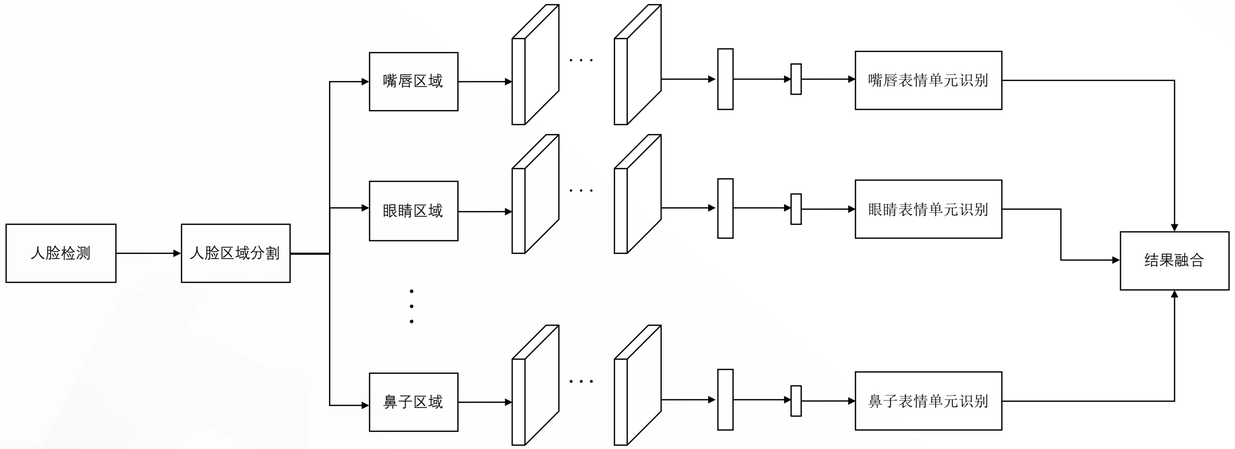
图书内容

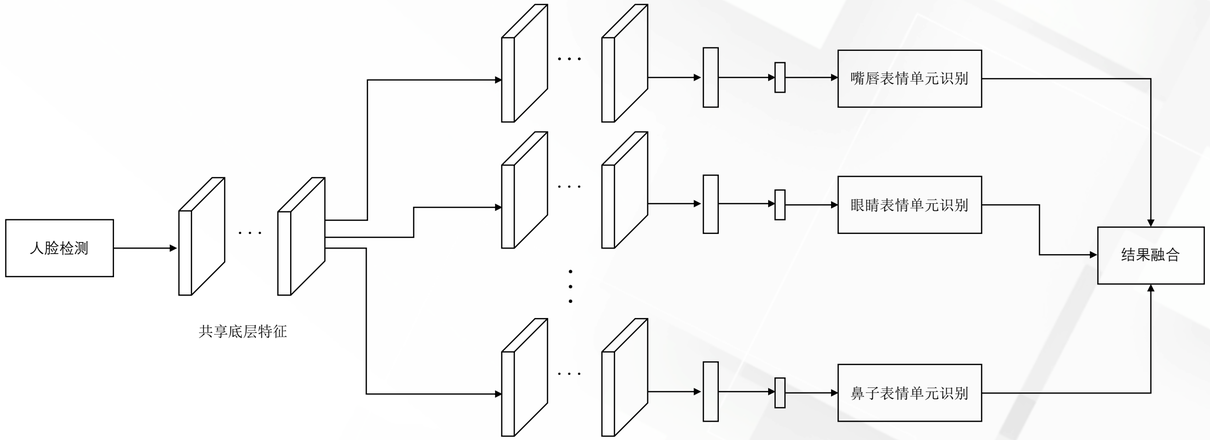
第七章：人脸属性识别

人脸性别识别，人脸颜值和脸型识别，人脸年龄识别，人脸表情识别，人脸表情和年龄属性识别项目实践







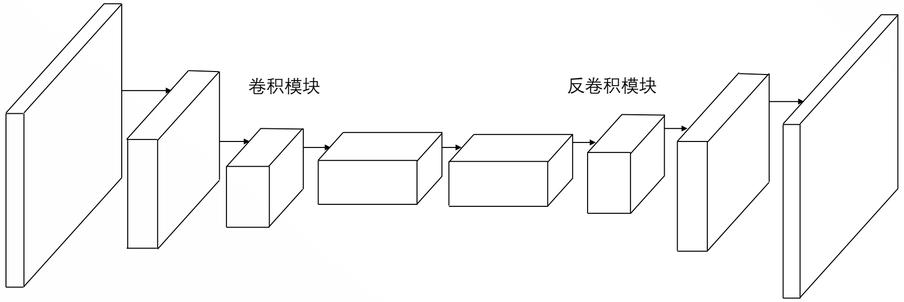


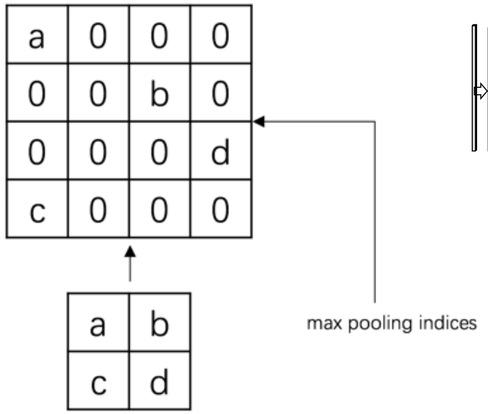
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

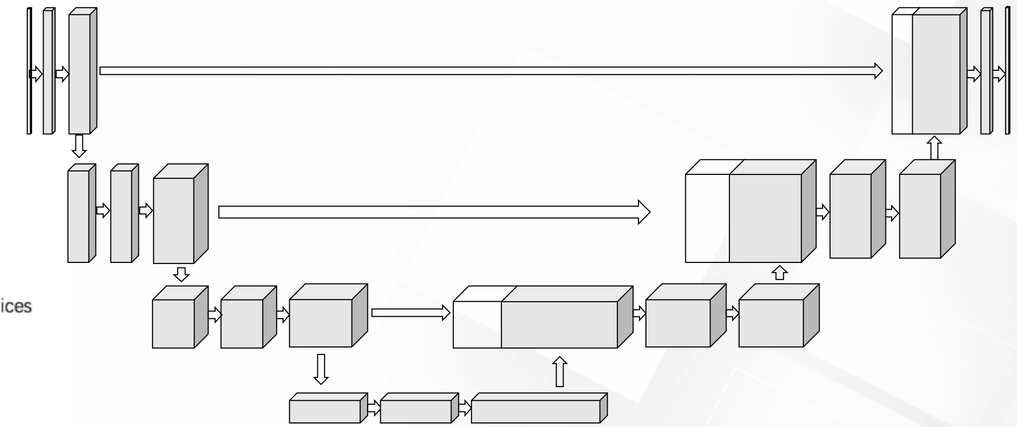
图书内容

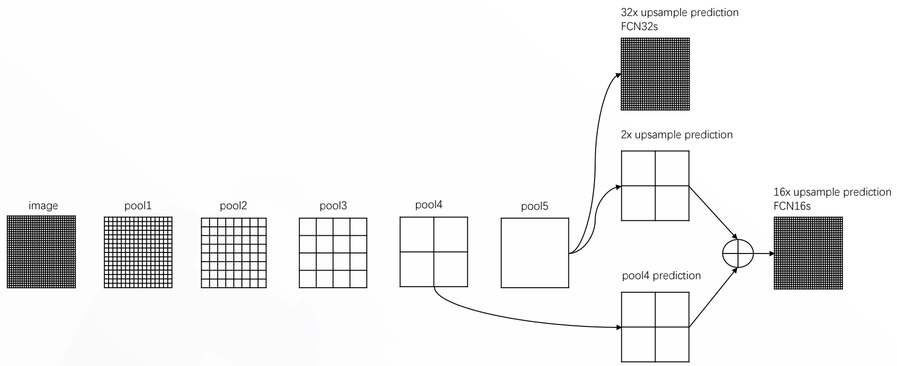
第八章：人脸属性分割

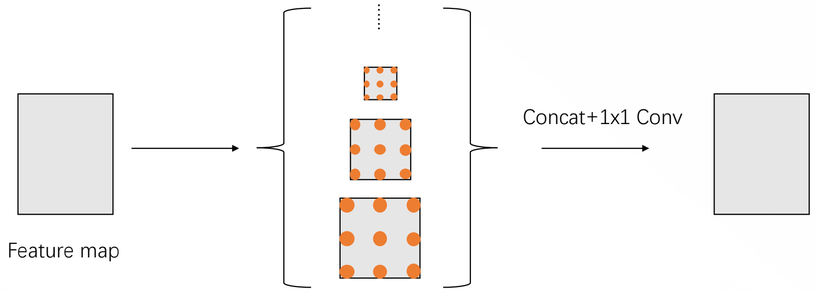
图像分割基础与人脸属性分割应用，深度学习图像分割核心技术，轻量级人脸分割项目实践

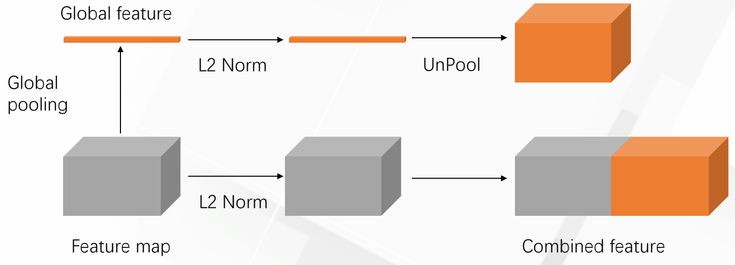










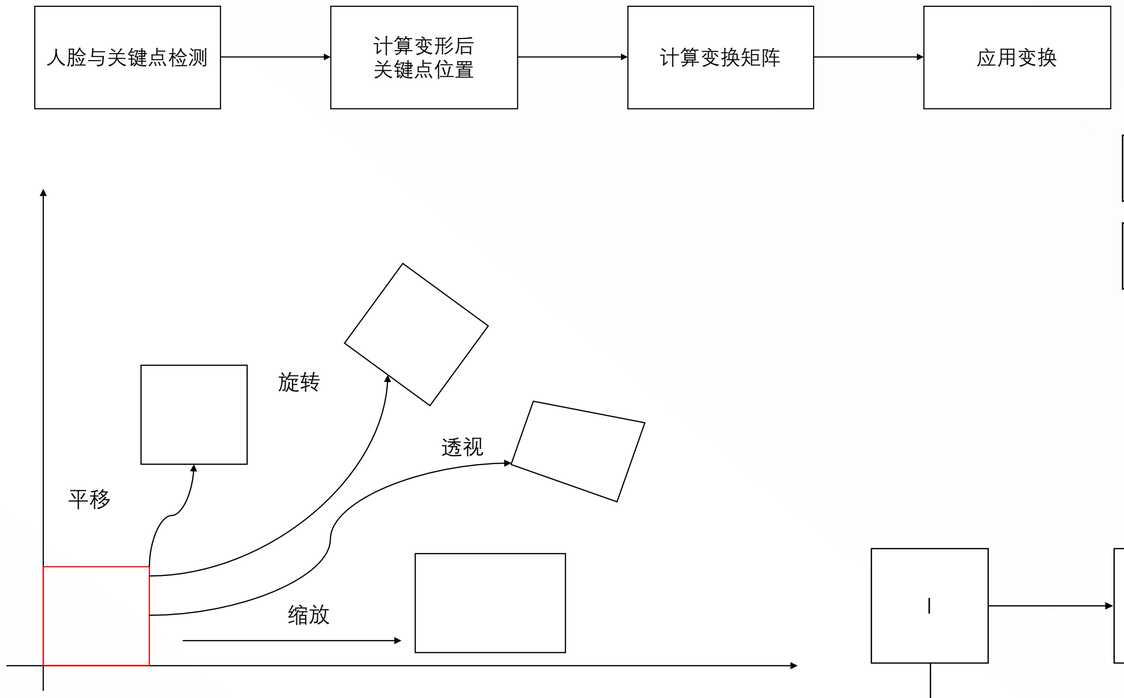


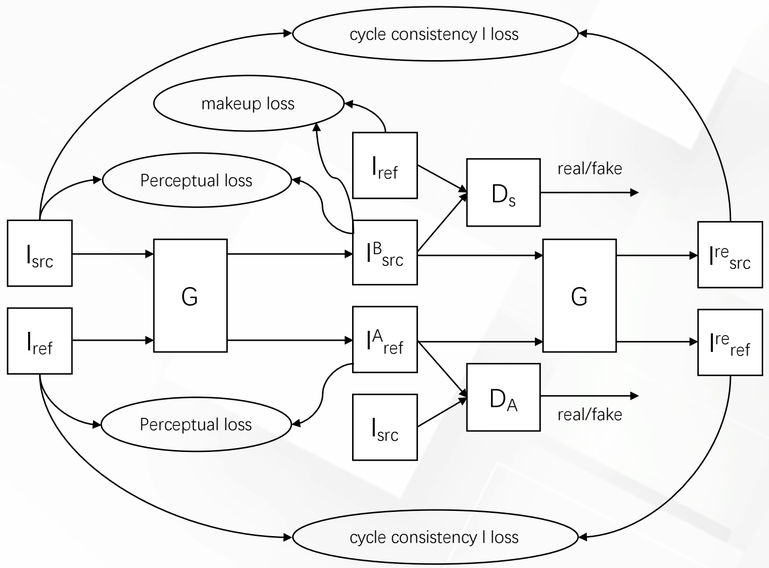
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

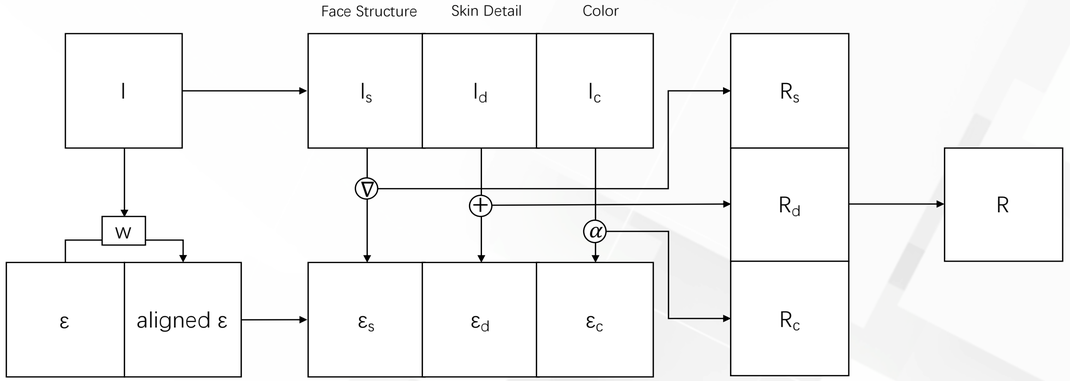
图书内容

第九章：人脸美颜

美颜基础和应用场景，滤波与变形美颜算法，妆造迁移算法，妆造迁移算法实战





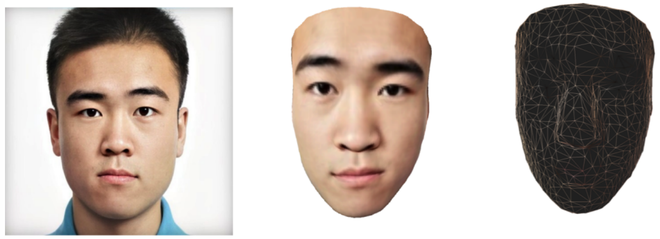


直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

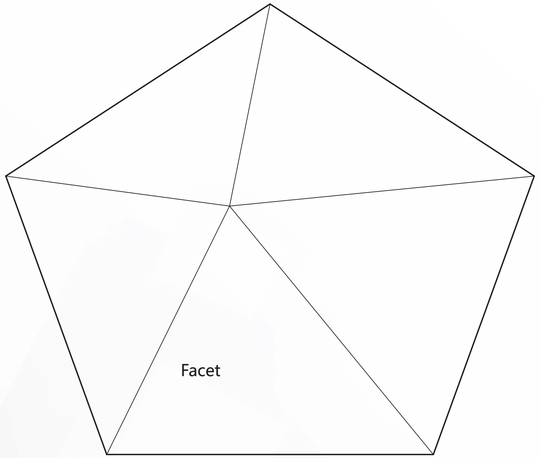
图书内容

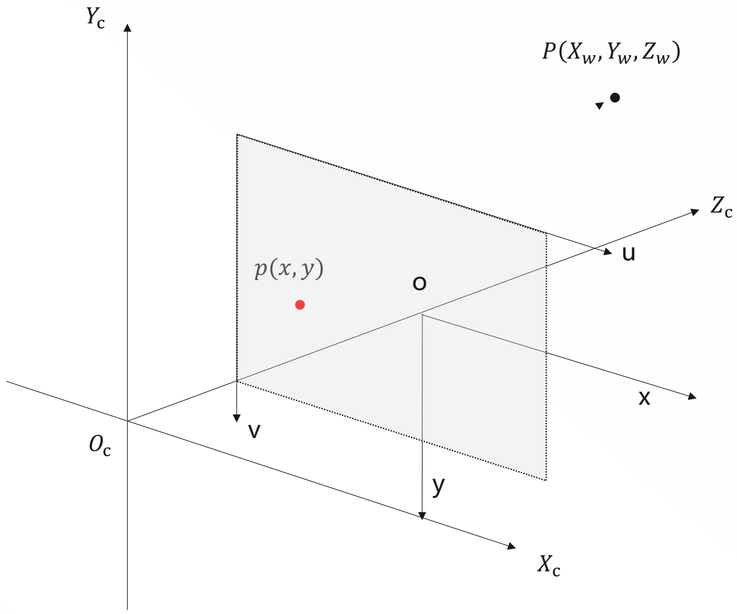
第十章：人脸三维重建

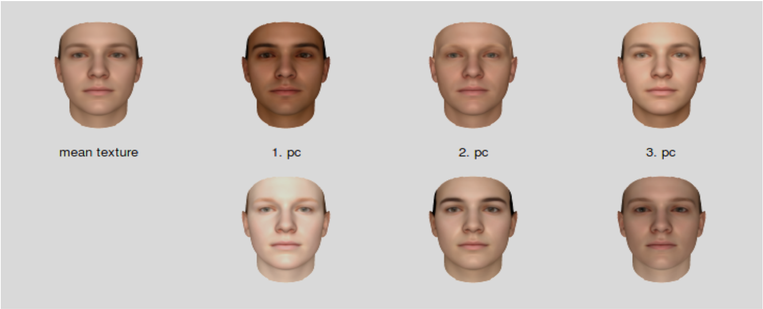
三维重建基础，传统三维人脸重建技术(立体视觉，3DMM，SFM，SfS)，深度学习三维人脸重建算法，深度学习三维人脸重建实践。

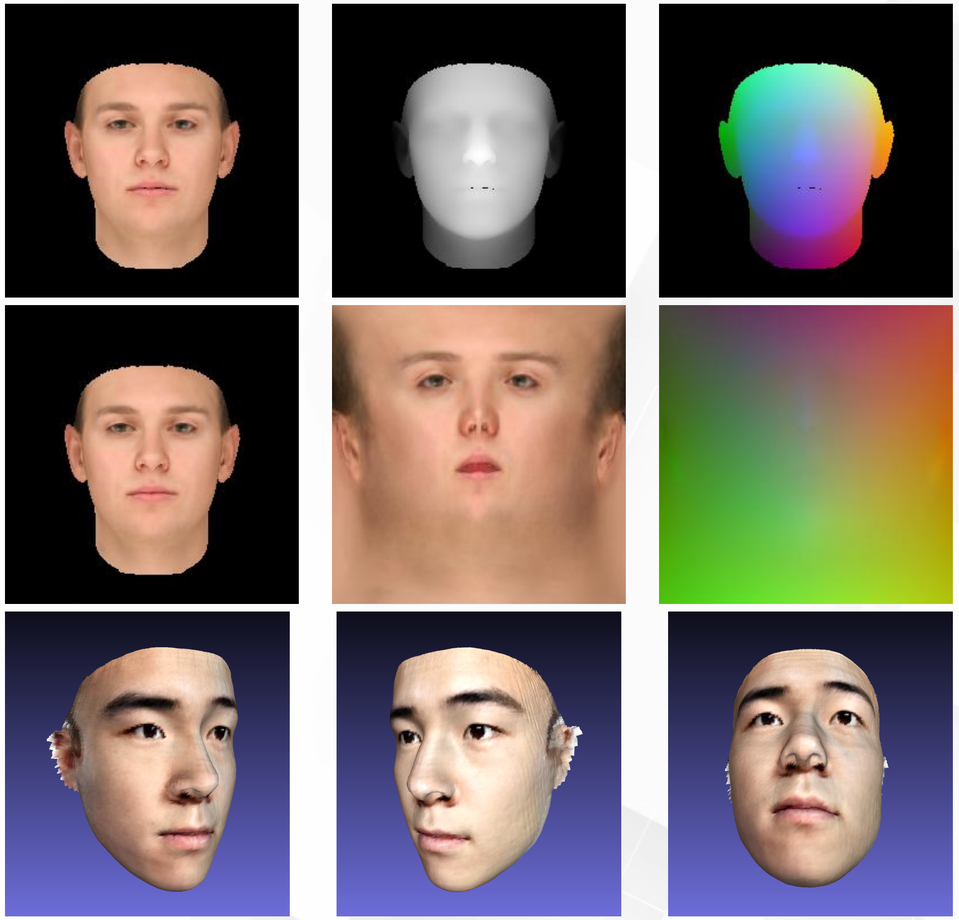


Vertex







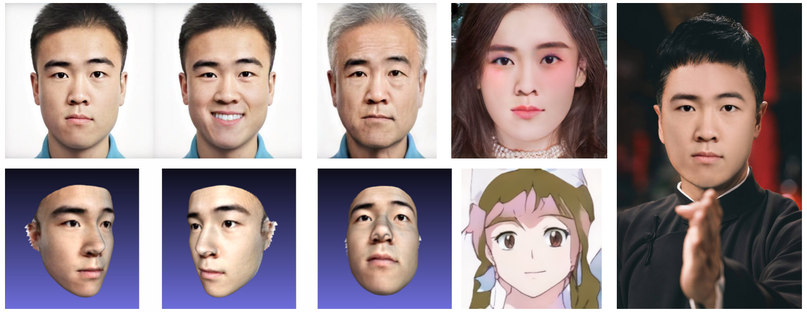


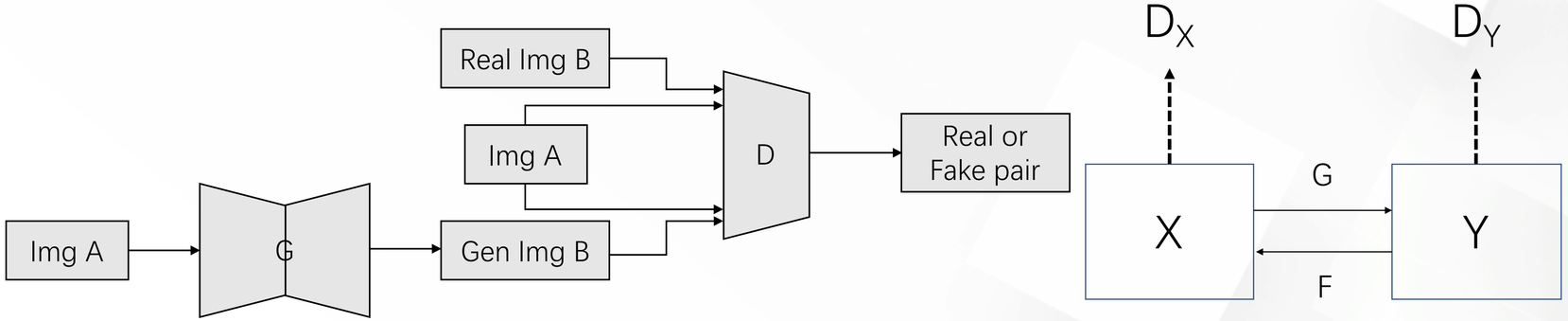
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

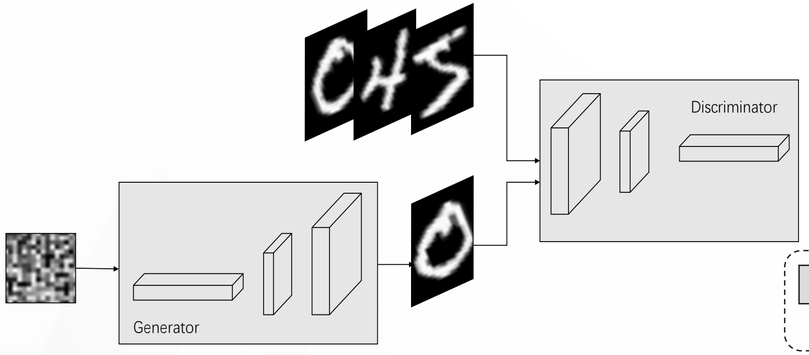
图书内容

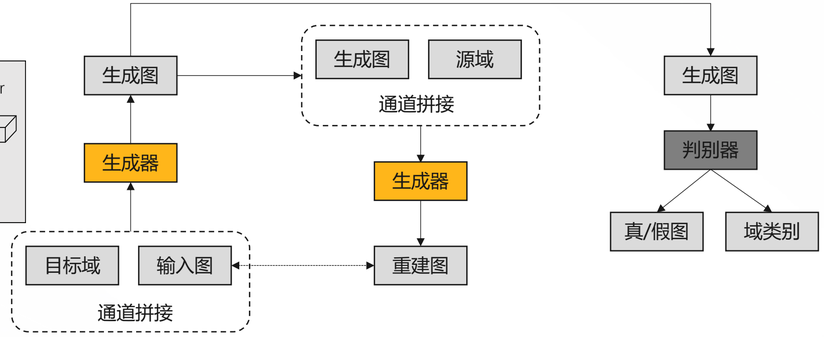
第十一章：人脸属性编辑

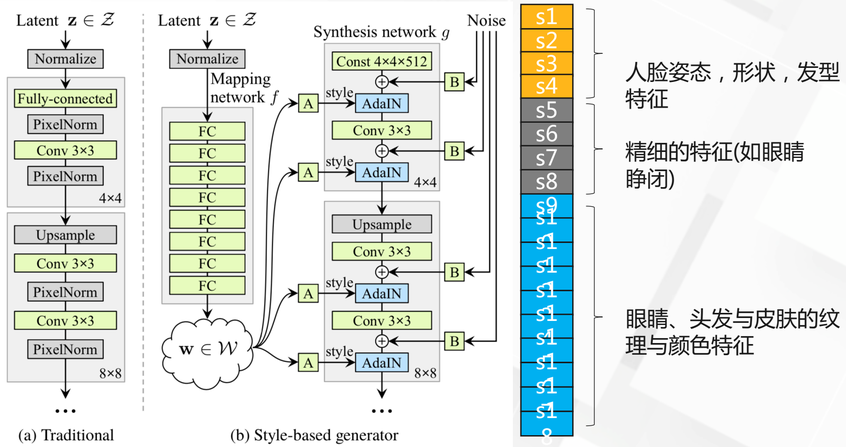
人脸属性编辑基础，深度学习人脸属性编辑方法，人脸动画头像风格化实战











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

- **内容全面**。本书以应用为脉络，详述了人脸检测，识别，属性分析与编辑，三维重建等方法，基本上覆盖了人脸图像算法的所有重要领域。不仅讲述了成熟的算法，也对研究还未成熟的领域进行了展望，如伪造脸检测，跨域人脸识别，无监督人脸重建模型等。

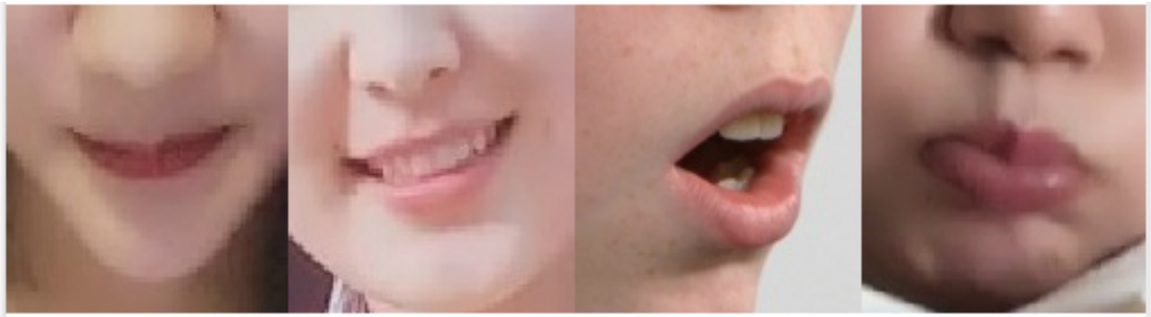
- **传统算法和深度学习算法兼具**。虽然本书名为《深度学习之人脸图像处理》，但是书中每一章都有一定的篇幅在讲述传统人脸图像处理算法，供大家拓展学习。

- **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1

TIANCHI天池

使用Pytorch搭建一个简易表情识别模型



任务解读：基于嘴唇的表情识别
应用场景：社交娱乐，安全控制，司法系统等

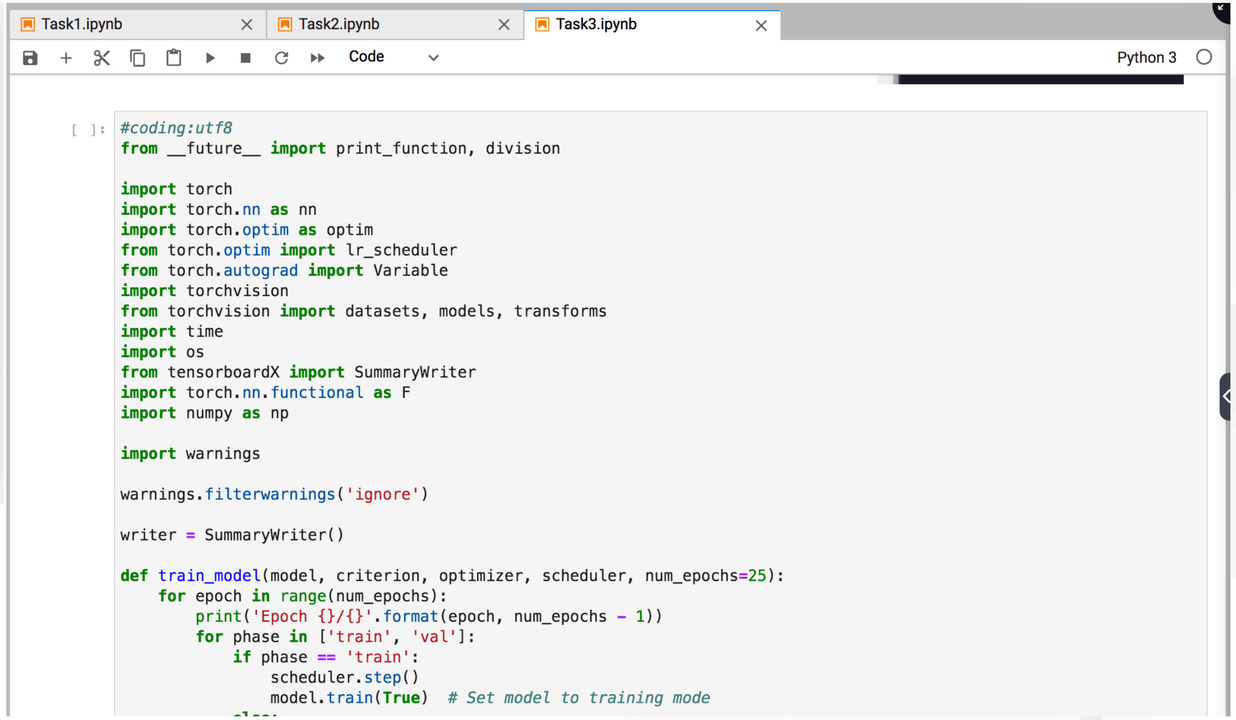
应用

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1

TIANCHI天池

项目入口： https://tianchi.aliyun.com/specials/promotion/activity/bookclub



Task1：深度学习理论简介
Task2：数据处理
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1

TIANCHI天池

完整的流程

选择开源学习框架 → **构建数据集并读取数据** → **网络模型选择** → **训练及参数调试** → **测试**

- Pytorch框架

- 爬虫、公开数据集
- 数据整理与预处理
- 撰写数据读取接口

- 定义分类模型

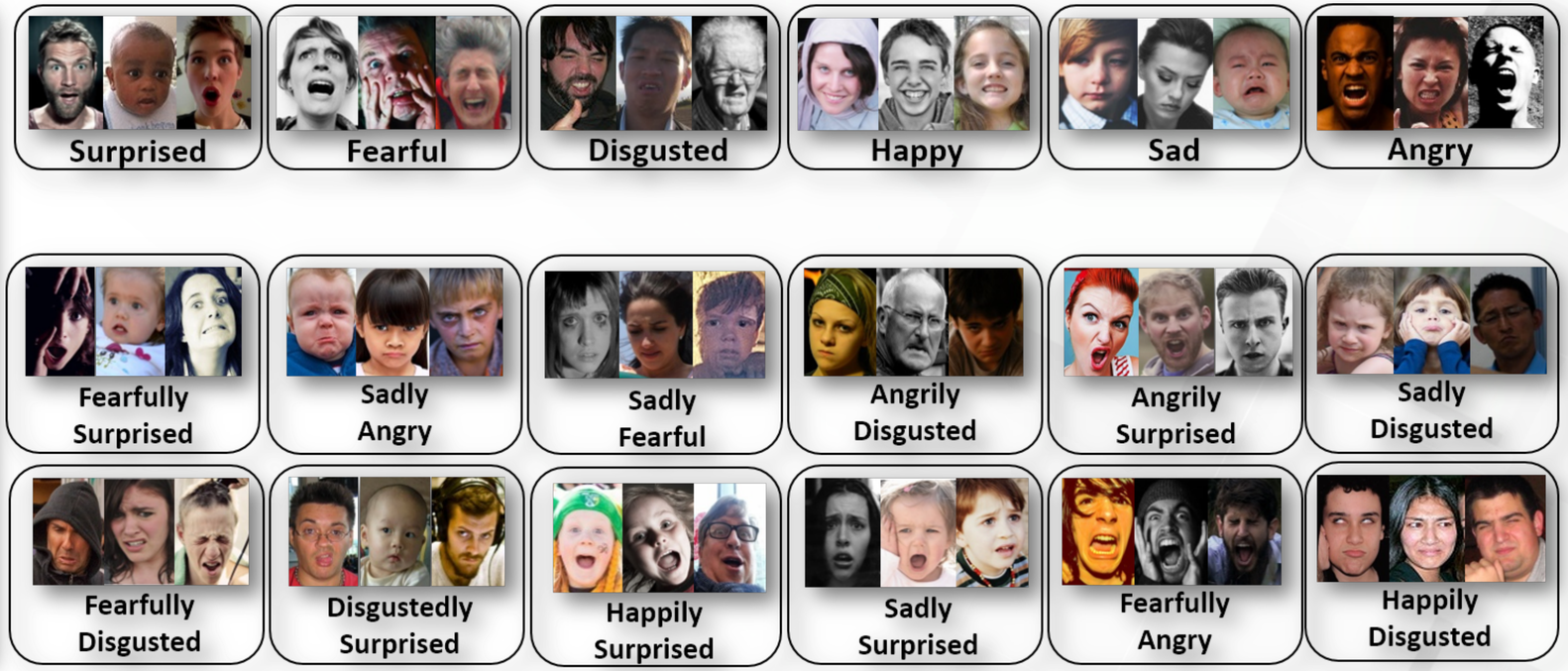
- 学习率
- 损失函数
- 优化方式
- 数据增强

- 模型推理

案例讲解1-数据获取

TIANCHI天池

表情数据集：KDEF,RaFD,RAF,EMotioNet等
人脸数据集：Celeba等

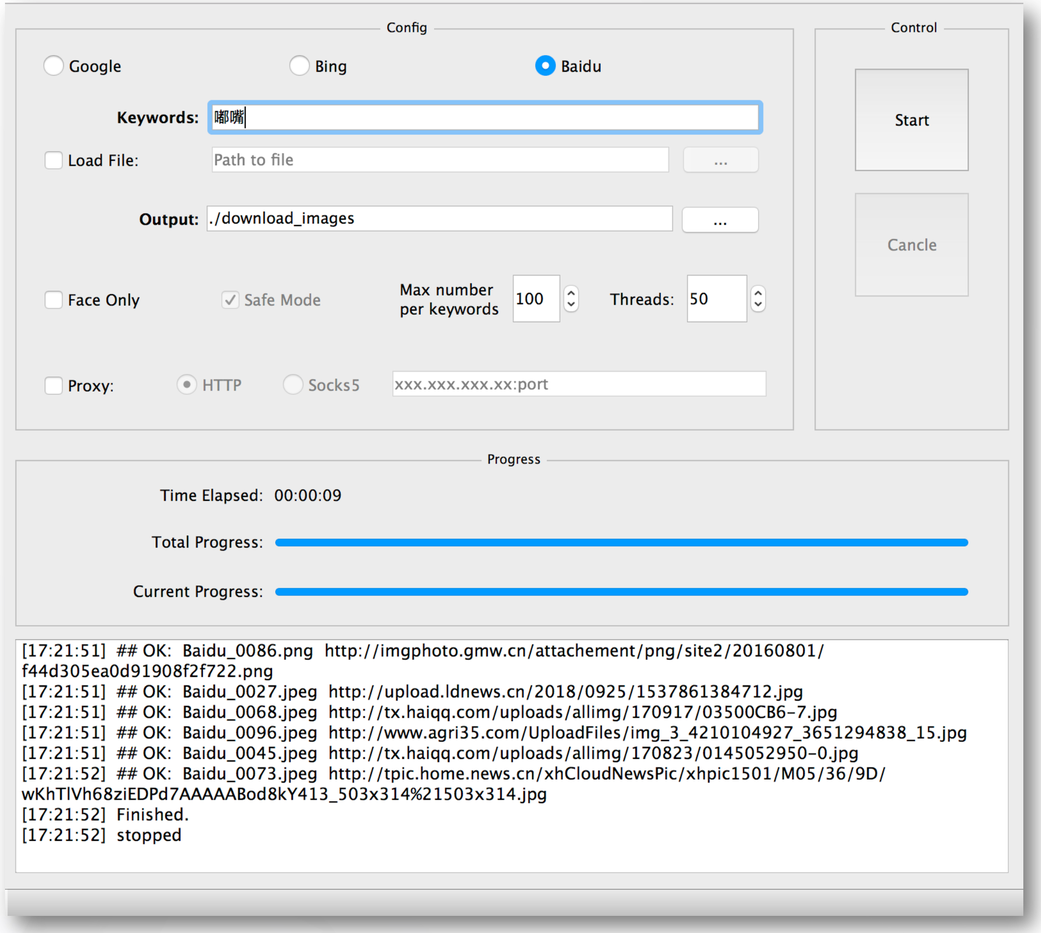


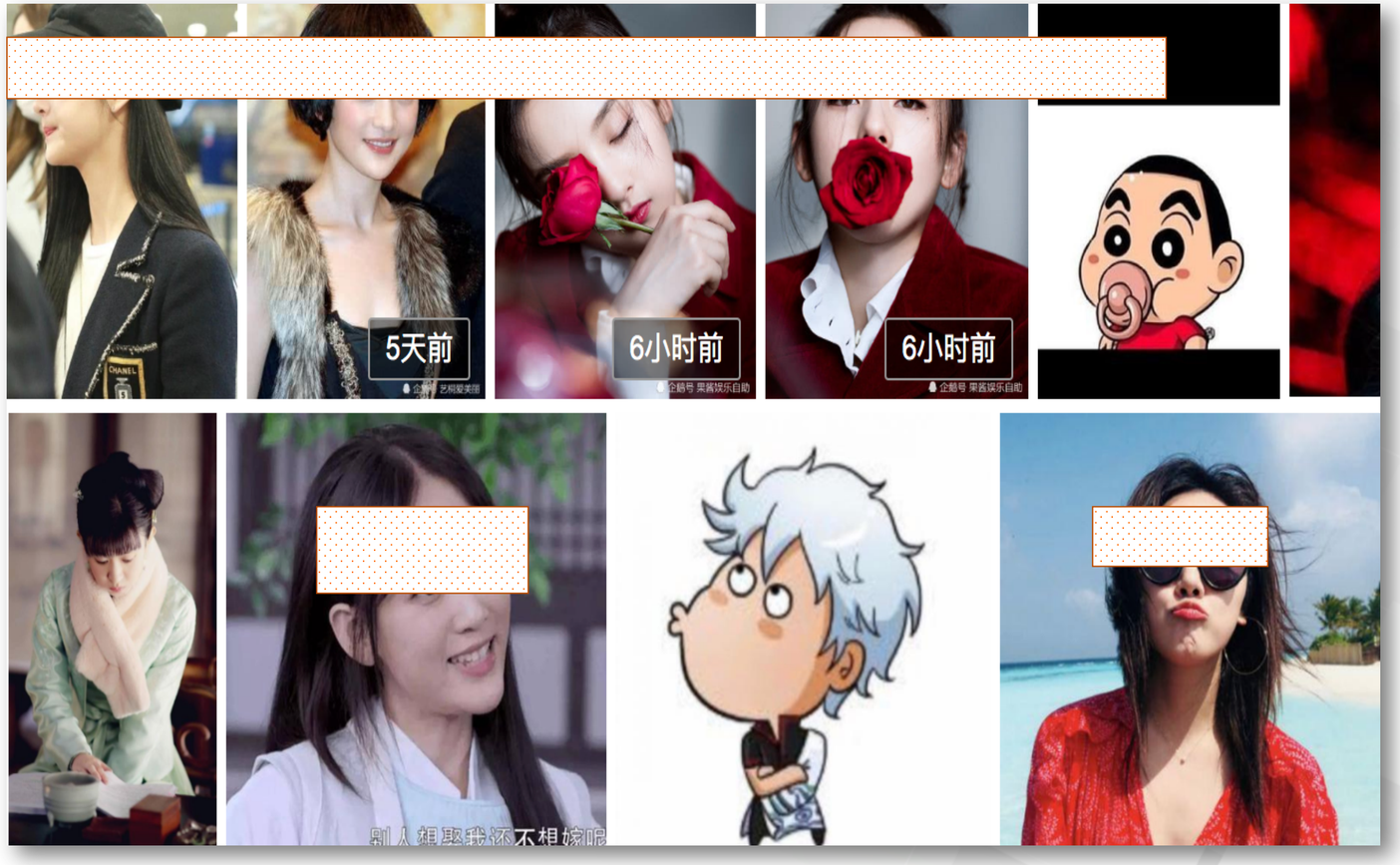
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

案例讲解1-数据获取

TIANCHI天池

https://github.com/sczhengyabin/Image-Downloader



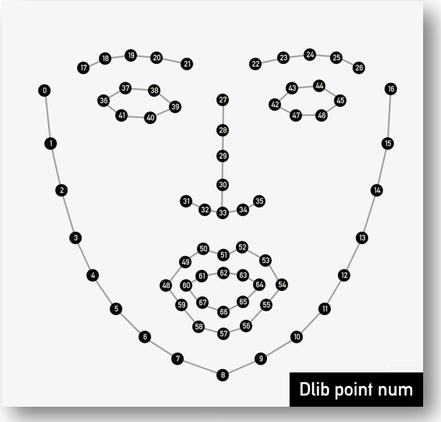


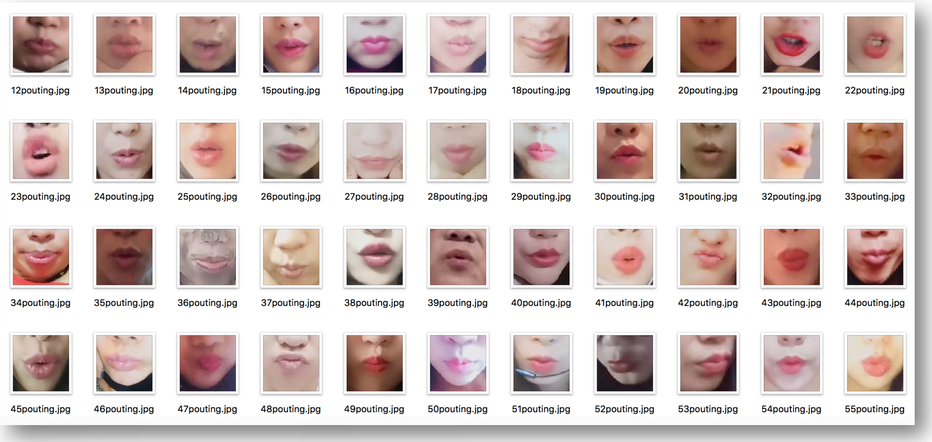
直播相关资料获取及回放查看地址：https://tianchi.aliyun.com/specials/promotion/activity/bookclub

案例讲解1-数据预处理

TIANCHI天池

OpenCV人脸检测，Dlib关键点检测
嘴唇区域裁剪并适当扩大区域





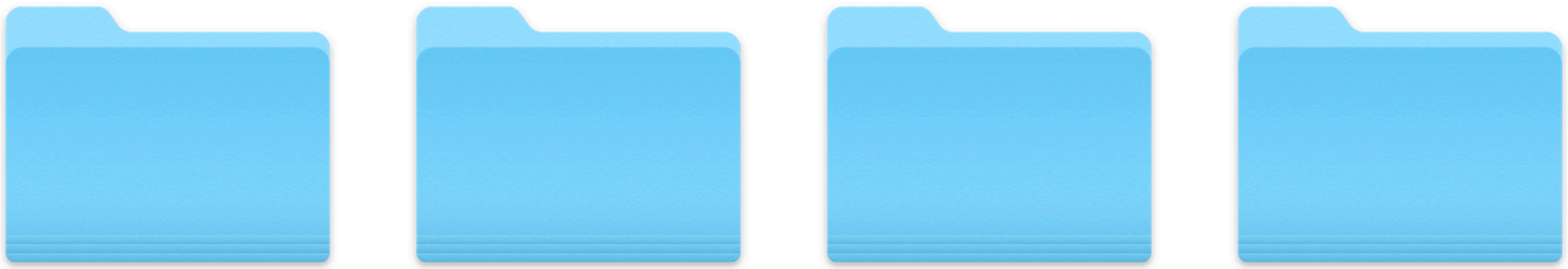

```
PREDICTOR_PATH = ".shape_predictor_68_face_landmarks.dat"
predictor = dlib.shape_predictor(PREDICTOR_PATH)
cascade_path = 'haarcascade_frontalface_default.xml'
cascade = cv2.CascadeClassifier(cascade_path)

def get_landmarks(im):
    rects = cascade.detectMultiScale(im, 1.3, 5)
    x, y, w, h = rects[0]
    rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
    return np.matrix([[p.x, p.y] for p in predictor(im, rect).parts()])

 for i in range(48, 67):
        x = landmarks[i, 0]
        y = landmarks[i, 1]
```

案例讲解1-数据集大小

15000多张图，包含微笑，嘟嘴，大笑，无表情4类，按照9:1均匀划分为训练集与测试集



0none　　1pouting　　2smile　　3openmouth

无表情0none：4763 训练集4287 测试集476
嘟嘴1pouting：3154 训练集2839 测试集315
微笑2smile：4841 训练集4357 测试集484
大笑3openmouth：2348 训练集2114 测试集234
训练数据：13597
测试数据：1509
总数据：15106
格式：统一为128*128，jpg图像

```
├── train
│   ├── 0
│   ├── 1
│   ├── 2
│   └── 3
└── val
    ├── 0
    ├── 1
    ├── 2
    └── 3
```

Pytorch图像分类任务文件夹格式

案例讲解1-数据接口

TIANCHI天池

Pytorch预处理：从64*64随机裁剪为48*48大小，去均值方差归一化

```
data_transforms = {
        'train': transforms.Compose([
            transforms.Scale(64),
            transforms.RandomSizedCrop(48),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize([0.5,0.5,0.5],
[0.5,0.5,0.5])
        ]),
        'val': transforms.Compose([
            transforms.Scale(64),
            transforms.CenterCrop(48),
            transforms.ToTensor(),
            transforms.Normalize([0.5,0.5,0.5],
[0.5,0.5,0.5])
        ]),
    }
```

```
 data_dir = './data/'
    image_datasets = {x:
datasets.ImageFolder(os.path.join(data_dir, x),
                                                 data_transforms[x]) for x in
['train', 'val']}
    dataloaders = {x:
torch.utils.data.DataLoader(image_datasets[x],
                                                  batch_size=16,
                                                  shuffle=True,
                                                  num_workers=4) for x in
['train', 'val']}
```

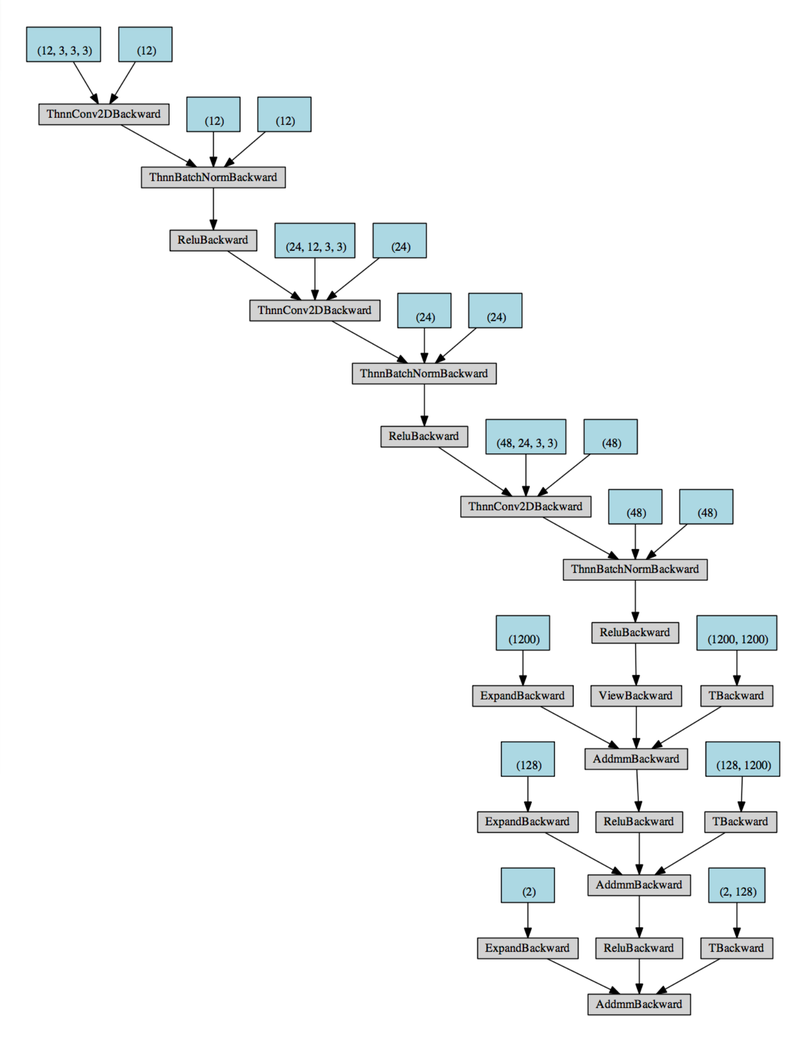
案例讲解1-网络定义

TIANCHI天池

定义1个3层的模型，由3个卷积层，3个BN层，3个全连接层构成

```
class simpleconv3(nn.Module):
    def __init__(self):
        super(simpleconv3,self).__init__()
        self.conv1 = nn.Conv2d(3, 12, 3, 2)
        self.bn1 = nn.BatchNorm2d(12)
        self.conv2 = nn.Conv2d(12, 24, 3, 2)
        self.bn2 = nn.BatchNorm2d(24)
        self.conv3 = nn.Conv2d(24, 48, 3, 2)
        self.bn3 = nn.BatchNorm2d(48)
        self.fc1 = nn.Linear(48 * 5 * 5 , 1200)
        self.fc2 = nn.Linear(1200 , 128)
        self.fc3 = nn.Linear(128 , 4)

    def forward(self , x):
        x = F.relu(self.bn1(self.conv1(x)))
        #print "bn1 shape",x.shape
        x = F.relu(self.bn2(self.conv2(x)))
        x = F.relu(self.bn3(self.conv3(x)))
        x = x.view(-1 , 48 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```



案例讲解1-优化目标与方法

TIANCHI天池

优化目标：交叉熵损失
优化方法：带动量的SGD算法(Momentum算法)

```
criterion = nn.CrossEntropyLoss() #交叉熵损失

optimizer_ft = optim.SGD(modelclc.parameters(), lr=0.1, momentum=0.9) #SGD优化方法
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=100, gamma=0.1) #学习率


#模型训练
modelclc = train_model(model=modelclc,
                       criterion=criterion,
                       optimizer=optimizer_ft,
                       scheduler=exp_lr_scheduler,
                       num_epochs=500)
```

案例讲解1-训练迭代

TIANCHI天池

每一次epoch，包含数个batch数据迭代

```
def train_model(model, criterion, optimizer, scheduler,
num_epochs=25):
    for epoch in range(num_epochs):
        for phase in ['train', 'val']:
            if phase == 'train':
                scheduler.step()
                model.train(True)
            else:
                model.train(False)

            running_loss = 0.0
            running_corrects = 0.0

            for data in dataloaders[phase]:
                inputs, labels = data
                if use_gpu:
                    inputs = Variable(inputs.cuda())
                    labels = Variable(labels.cuda())
```

```
else:
    inputs, labels = Variable(inputs), Variable(labels)

optimizer.zero_grad()
outputs = model(inputs)
_, preds = torch.max(outputs.data, 1)
loss = criterion(outputs, labels)
if phase == 'train  ':
    loss.backward()
    optimizer.step()
```

每一个迭代：optimizer.zero_grad()

案例讲解1-可视化

TIANCHI天池

借用tensorboardX

```
from tensorboardX import SummaryWriter
writer = SummaryWriter()
```

```
# 每一个batch
if phase == 'train  ':
    writer.add_scalar('data/trainloss', epoch_loss, epoch)
                writer.add_scalar('data/trainacc', epoch_acc, epoch)
else:
    writer.add_scalar('data/valloss', epoch_loss, epoch)
    writer.add_scalar('data/valacc', epoch_acc, epoch)
```

```
# 每一个epoch
    writer.export_scalars_to_json("./all_scalars.json")
    writer.close()
```

案例讲解1-测试

TIANCHI天池

载入预训练模型，前向推理获得预测结果

```
from net import simpleconv3
# 定义模型并载入权重
net = simpleconv3()
net.load_state_dict(torch.load(modelpath,map_location=lambda storage,loc: storage))
net.eval() #设置为推理模式，不会更新模型的k，b参数
```

```
# 定义数据预处理
testsize = 48 ##测试图大小
data_transforms =  transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize([0.5,0.5,0.5], [0.5,0.5,0.5])])
```

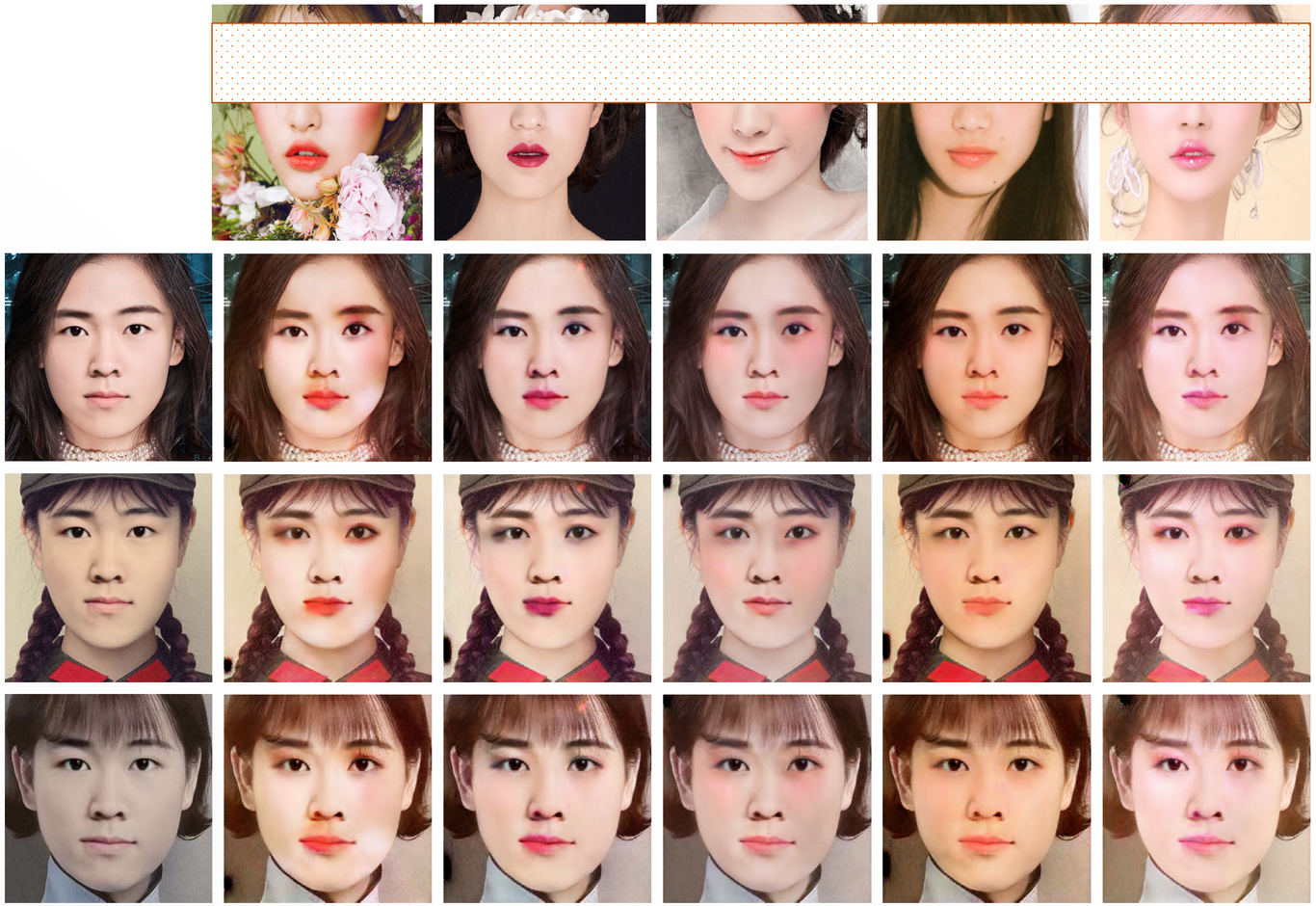
```
# 前向推理获得结果
torch.no_grad() #停止autograd模块的工作，加速和节省显存
imgblob = data_transforms(img).unsqueeze(0) #填充数据维度
imgblob = Variable(imgblob)
predict = F.softmax(net(imgblob))
index = np.argmax(predict.detach().numpy())
```

```
# 绘制到图像
Opencv的rectangle函数与putText函数
```

案例讲解2

TIANCHI天池

使用Pytorch训练一个人脸美妆模型（第9章）

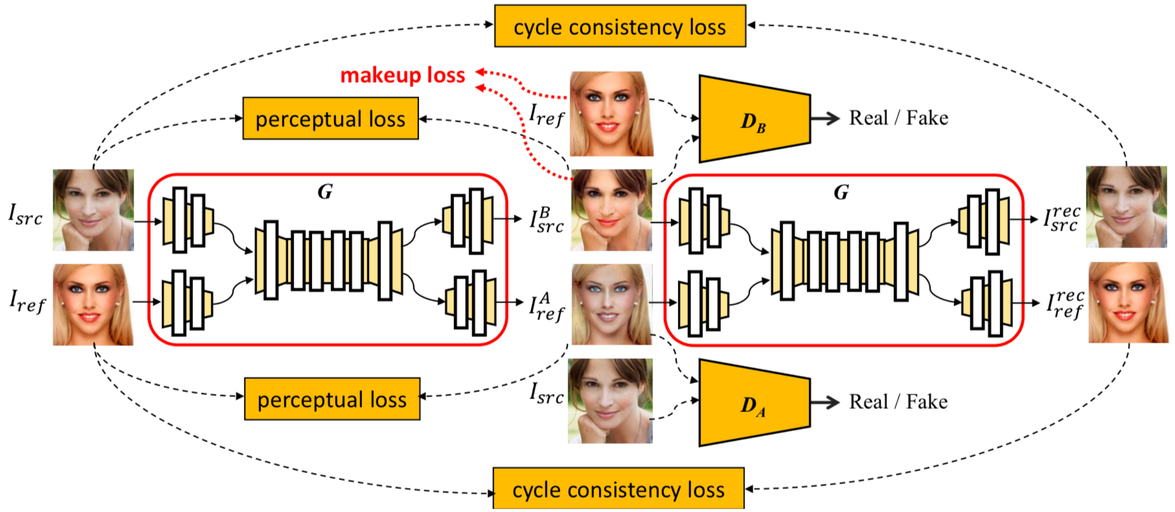


任务背景：自适应的妆造迁移
核心技术：CycleGAN模型变种

案例讲解2

模型架构：BeautyGAN模型，输入两张人脸图片，一张无妆图I_{src} ，一张有妆图I_{ref} ，模型输出换妆之后的结果，即一张上妆图I^B_{src}和一张卸妆图I^A_{ref} 。

经典的无监督循环图像翻译结构： $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$



正向过程：生成器G输入无妆图I_{src}、有妆图I_{ref}，通过编码器(encoder)、若干个残差模块(residual blocks)、解码器(decoder)组成的生成器G 得到两个输出，分别是上妆图I^B_{src}、卸妆图I^A_{ref}

反向过程：上妆图I^B_{src}和卸妆图I^A_{ref}再次输入给G，重新执行一次卸妆和上妆，得到两张重建图I^{re}_{src}和卸妆图I^{re}_{ref}

$$\mathcal{L}_{cyc} = \mathbb{E}_{I_{src}, I_{ref}}[dist(I^{rec}_{src}, I_{src}) + dist(I^{rec}_{ref}, I_{ref})]$$

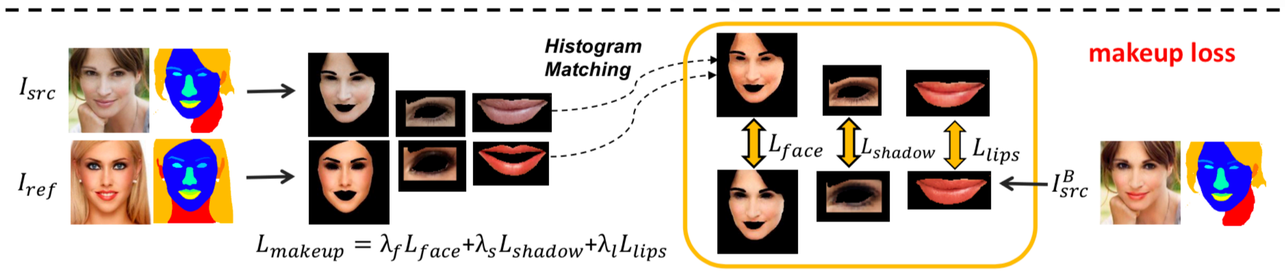
L1距离

$$(I^{rec}_{src}, I^{rec}_{ref}) = G(G(I_{src}, I_{ref}))$$

案例讲解2

TIANCHI天池

局部的风格迁移损失



使用语义分割网络提取人脸不同区域的掩膜(mask)，使得无妆图和有妆图在脸部、眼部、嘴部三个区域需满足妆造损失(makeup loss)，妆造损失通过直方图匹配实现，其中一个区域的损失定义如下：

$$L_{item} = ||I^B_{src} - HM(I^B_{src} \circ M^1_{item}, I_{ref} \circ M^2_{item})||^2_2$$

M^1_{item}，M^2_{item}分别表示两个I^B_{src}和I_{ref}对应的区域掩膜，$\circ$表示逐像素的相乘，$item$可以分别表示脸部、眼部、嘴部三个区域，HM是一个直方图匹配操作。

$$\mathcal{L}_{makeup} = \lambda_l \mathcal{L}_{lips} + \lambda_s \mathcal{L}_{shadow} + \lambda_f \mathcal{L}_{face}$$

风格迁移损失

案例讲解2

TIANCHI天池

保持身份的感知损失信息，上妆和卸妆不能改变原始的人物可识别身份信息

基于VGG模型的Perceptual loss进行约束，$L_{per} = \frac{1}{C_l \times H_l \times W_l} \sum_{ijk} E_l$

其中C_l，H_l，W_l分别是网络第l层的通道数，特征图高度和宽度，E_l是特征的欧式距离，包含两部分如下：

$$E_l = [F_l(I_{src}) - F_l(I^B_{src})]^2_{ijk} + [F_l(I_{ref}) - F_l(I^A_{ref})]^2_{ijk}$$

案例讲解2

TIANCHI天池

生成器代码示意

```
class Generator_makeup(nn.Module):
    # 生成器是一个编解码结构，输入两张图，输出两张图
    def __init__(self, conv_dim=64, repeat_num=6, input_nc=6):
        super(Generator_makeup, self).__init__()

        layers = []
        ## 第一个卷积层，输入维度为6，它是无妆造图像和有妆造图通过通道拼接，卷积核大小kernel_size为7，步长stride为1，填充边界像素宽padding为3
        layers.append(nn.Conv2d(input_nc, conv_dim, kernel_size=7, stride=1, padding=3, bias=False))
        layers.append(nn.InstanceNorm2d(conv_dim, affine=True)) ##InstanceNorm层
        layers.append(nn.ReLU(inplace=True)) ##ReLU层

        # 下采样+上采样编码器模块
        .......
        self.main = nn.Sequential(*layers) ##主干通道输出
```

```
        ##接下来是两个分支的定义，当输入为原图和妆造图时，分支1输出原图的妆造图，分支2输出为妆造图的卸妆图，当输入为原图化妆后的图和妆造卸妆后的图时，分支1输出卸妆图，分支2输出为妆造图

        ##分支1定义，包含一个7*7的卷积层和一个tanh激活函数层
        layers_1 = []
        layers_1.append(nn.Conv2d(curr_dim, 3, kernel_size=7, stride=1, padding=3, bias=False))
        layers_1.append(nn.Tanh())
        self.branch_1 = nn.Sequential(*layers_1)

        layers_2 = [
        layers_2.append(nn.Conv2d(curr_dim, 3, kernel_size=7, stride=1, padding=3, bias=False))
        layers_2.append(nn.Tanh())
        self.branch_2 = nn.Sequential(*layers_2)
```

案例讲解2

TIANCHI天池

判别器代码示意（使用了PatchGAN ）

```
class Discriminator(nn.Module):
    def __init__(self, image_size=128, conv_dim=64,
repeat_num=3)
        super(Discriminator, self).__init__()
        layers = []

        ## 第一个卷积层定义，输入为3通道图像，卷积核大小为4×4，
步长为2
        layers.append(nn.Conv2d(3, conv_dim, kernel_size=4,
stride=2, padding=1))
        layers.append(nn.LeakyReLU(0.01, inplace=True))

        ## 重复repeat_num个卷积层定义，每一个卷积核大小为4×4，
步长等于2，输出通道为输入的两倍
        curr_dim = conv_di
        for i in range(1, repeat_num):
            layers.append(nn.Conv2d(curr_dim, curr_dim*2,
kernel_size=4, stride=2, padding=1))
            layers.append(nn.LeakyReLU(0.01, inplace=True))
            curr_dim = curr_dim * 2
```

```
        # 主干模型最后一个卷积层定义，卷积核大小为4×4，
步长为1
        layers.append(nn.Conv2d(curr_dim, curr_dim*2,
kernel_size=4, stride=1, padding=1))

        layers.append(nn.LeakyReLU(0.01, inplace=True))

        curr_dim = curr_dim *2

        self.main = nn.Sequential(*layers)

        # 输出卷积层定义，卷积核大小为4×4，步长为1
        self.conv1 = nn.Conv2d(curr_dim, 1,
kernel_size=4, stride=1, padding=1, bias=False)
```

如何补充理论知识

TIANCHI天池

阿里天池&有三AI深度学习课程
https://tianchi.aliyun.com/course/279

课程目录

深度学习

- 第1课：人工智能简介
- 第2课：深度学习崛起背景
- 第3课：深度学习典型应用与研究方向之语音处理
- 第3课：深度学习典型应用与研究方向之计算机视觉
- 第3课：深度学习典型应用与研究方向之自然语言处理 00:22:27
- 第3课：深度学习典型应用与研究方向之推荐系统 00:20:40
- 第4课：神经网络（上） 00:33:03
- 第4课：神经网络（下） 00:32:48
- 第5课：卷积神经网络（上） 00:42:53
- 第5课：卷积神经网络（下） 00:29:42
- 第6课：激活函数与参数初始化 00:41:55
- 第6课：标准化与池化 00:29:37

课程内容将包括人工智能与深度学习发展背景，深度学习典型应用，卷积神经网络，循环神经网络，生成对抗网络，深度学习开源框架等内容，预计共40余课时

阿里云天池&有三AI-深度...
37人

扫一扫群二维码，立刻加入该群。

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

实战演示、互动交流

TIANCHI天池

大家可以使用**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 **动手实践**

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

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