计算机视觉作业 7

信息工程学院 张瀚文 2201212865

1. 作业要求

运行一个超分算法，训练一两个 Epoch，给出超分结果。

算法 Reference: https://github.com/zzxvictor/License-super-resolution

Author: Zixuan Zhang, Chengxuan Cai

本次作业

北京大学 PEKING UNIVERSITY

Github或者主页下载运行一个超分算法，获得结果试着训练一两个Epoch，给出超分结果

2. 实现过程

2.1 车牌增强超分算法

车牌增强是单图像超级分辨率（SISR）的更广泛领域的详细应用。

该项目受到了以下几种最先进 SR 模型的启发：

- Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network
- Residual Dense Network for Image Super-Resolution
- ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks
- Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network

该项目使用的数据集来自：https://github.com/detectRecog/CCPD，可以直接下载使用。

2.2 安装项目依赖库

数据预处理：

- Dask >= 2.11.0
- PIL >= 6.2.2

训练与评估：

- tensorflow >= 2.1.0
- numpy >= 1.18.1
- matplotlib >= 3.1.3

2.3 训练过程

2.3.1 数据预处理

首先从 https://github.com/detectRecog/CCPD 下载数据集，然后运行 preprocess.py 文件进行数据预处理。方法：在终端输入 python preprocess.py 5 PATH_TO_UNZIPPED_DATA PATH_TO_OUTPUT_DIR 指令。

In [2]:
```
from Utilities.io import DataLoader
from Utilities.lossMetric import *
from Utilities.trainVal import MinMaxGame
from Models.RRDBNet import RRDBNet
from Models.GAN import Discriminator
```

Load in the training dataset

I used the Chinese City Parking Dataset for this project. Please download the dataset from https://github.com/detectRecog/CCPD
Before loading the dataset, it is critical that you run the preprocessing script (preprocess.py) first!!! `python preprocess.py 5 PATH_TO_UNZIPPED_DATA PATH_TO_OUTPUT_DIR`

In [8]:
```
import numpy as np
import glob
PATH = 'PATH_TO_OUTPUT_DIR/192_96' # only use images with shape 192 by 96 for training
files = glob.glob(PATH + '/*.jpg') * 3  # data augmentation, same image with different brightness and contrast
np.random.shuffle(files)
train, val = files[:int(len(files)*0.8)], files[int(len(files)*0.8):]
loader = DataLoader()
trainData = DataLoader().load(train, batchSize=16)
valData = DataLoader().load(val, batchSize=64)
```

2.3.2 训练 SR 模型

设定 Epoch=1，尝试训练 SR 模型。

Training

In [9]:
```
discriminator = Discriminator()
extractor = buildExtractor()
generator = RRDBNet(blockNum=10)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 8s 0us/step
```

- It's a good idea to pretrain the generator model before the min-max game - Reference: https://arxiv.org/abs/1701.00160

In [10]:
```
# a simple custom loss function that combines MAE loss with VGG loss, as defined in the SRGAN paper
def contentLoss(y_true, y_pred):
    featurePred = extractor(y_pred)
    feature = extractor(y_true)
    mae = tf.reduce_mean(tfk.losses.mae(y_true, y_pred))
    return 0.1*tf.reduce_mean(tfk.losses.mse(featurePred, feature)) + mae

optimizer = tfk.optimizers.Adam(learning_rate=1e-3)
generator.compile(loss=contentLoss, optimizer=optimizer, metrics=[psnr, ssim])
# epoch is set to 1 for demonstration purpose. In practice I found 20 is a good number
# When the model reaches PSNR=20/ssim=0.65, we can start the min-max game
history = generator.fit(x=trainData, validation_data=valData, epochs=1, steps_per_epoch=3, validation_steps=1)
```

```
3/3 [==============================] - 22s 8s/step - loss: 8.2637 - psnr: 2.1568 - ssim: 0.0106 - val_loss: 5.1731
- val_psnr: 4.6175 - val_ssim: 0.0078
```

2.3.3 训练 GAN 增强模型

Generative adverserial network training

In [11]:
```
# training parameter. epoch is set to 1 for demonstration
# please train the network utill it reaches snRatio ~= 22
PARAMS = dict(lrGenerator = 1e-4,
              lrDiscriminator = 1e-4,
              epochs = 1,
              stepsPerEpoch = 500,
              valSteps = 100)
game = MinMaxGame(generator, discriminator, extractor)
log, valLog = game.train(trainData, valData, PARAMS)
# ideally peak signal noise ratio(snRation or psnr) should reach ~22
```

3. 实验结果

使用作者训练好的模型，做超分车牌增强处理。

3.1 导入预训练模型

Load in the pretrained super-resolution model

In [3]:
```
# pretrained rrdb network can be found in the Pretrained folder
MODEL_PATH = 'Pretrained/rrdb'
model = RRDBNet(blockNum=10)
model.load_weights(MODEL_PATH)
```

Out[3]:
```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7ff601906040>
```

3.2 测试模型

Run plate enhancement

In [4]:
```
for downSample, original in data.take(4):
    yPred = model.predict(downSample)
    painter.plot(downSample, original, yPred)
```

```
1/1 [==============================] - 1s 679ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 64ms/step
```