DSLR-Quality Photos on Mobile Devices with Deep Convolutional Networks 经典论文复现与改进

摘要： 图像增强是当下火热技术之一，我们以该领域经典方法[1]为基础，在了解相关技术之外，逐一使用多种方法去尝试改进，如 Batch Normalization、Cycle-GAN 网络、Densenet 网络生成器等。本文对经典方法进行了简单介绍并记录了各种尝试的过程，最后总结并给出自己的思考。

关键字： 正则化，生成对抗网络，卷积神经网络，Unpaired 训练

一. 经典论文概述

在 DSLR-Quality Photos on Mobile Devices with Deep Convolutional Networks 一文中（下称 DPED），作者提出了一种基于卷积神经网络的方法去实现将手机拍摄的低画质图像转换成单反拍摄的高画质图像。

我们认为这篇文章最为宝贵之处在于提出了四种损失函数，包含 Color-Loss, Content-Loss, Texture-Loss 以及 Total-Variation-Loss，其中 Texture-Loss 作为精髓，采用了生成对抗网络的形式，关于它们的简介如下所列。

- **Color-Loss**：用于评估生成器生成的伪单反图片与真实图片色彩差异，在实现中，首先对伪图和真图进行高斯模糊化（高斯卷积层处理），然后计算欧式距离作为损失函数。
- **Content-Loss**：用于衡量生成器生成的伪单反图片与真实图片内容差异，在实现中，将两类图片分别送入预训练的 VGG-19 网络，正向传播时取某一 ReLu 激活输出作为损失函数。
- **Total-Variation-Loss**：用于控制生成器生成图片整体的平滑性，在实现中，以图片正则化后的行梯度和列梯度之和作为损失函数，通常这个值较小，需要更大的权重使之生效。
- **Texture-Loss**：用于比较伪图和真实图片的纹理差异，其采用了 GAN 网络结构，具体来说，文章采用 Resnet 作为生成器，用于将手机拍摄的照片转化为高画质单反图片，文章中增加了一个普通的卷积神经网络作为判别器去评判生成图片真伪与否，于是 Resnet 和该卷积网络形成生成对抗关系，最终能够提高生成效果，而判别器输出的交叉熵函数作为整个网络结构的纹理-损失函数。

将上述损失函数用于如图 1 所示的网络结构中，通过训练该神经网络，最终我们将得到一个较好的生成器用于图像增强。

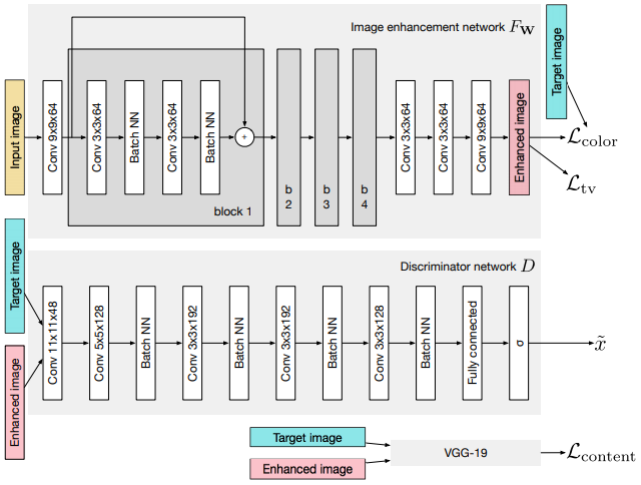

图 1. DPED 论文网络结构

二. 经典论文复现

感谢论文原作者公开了具体的实现细节，我们从论文主页获得了相关数据与源代码，下面是在 blackberry 数据集上的测试结果。

图二展示了训练过程中训练集与测试集损失函数和判别器准确率的波动曲线，从图中我们发现判别器的准确率在 0.5 附近波动，符合 GAN 网络的优化情景，而所有损失函数之和如右图所示，随着迭代次数的增加，呈现下降的趋势。

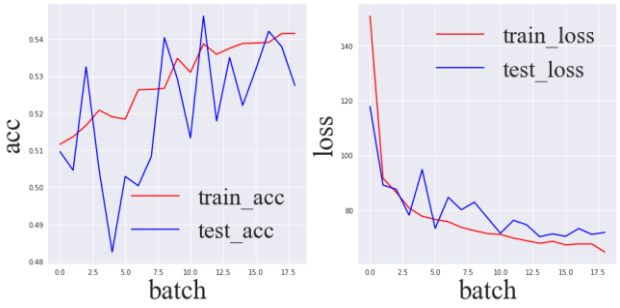

图 2. DPED 论文复现结果

三. NORMALIZATION 层的选择

虽然作者在原文中提及网络结构中层与层之间的 Normalization 层使用了 Batch Normalization，但是在阅读源代码的过程中我们发现它大部分使用的还是 Instance Normalization，下文是两种 Normalization 方式的原理介绍，在实验中，我们尝试将其改为 BN 方式并记录训练过程，最后给出测试结果与思考分析。

3.1 Batch Normalization

[1] *DSLR-Quality Photos on Mobile Devices with Deep Convolutional Networks*

BN 是由 Google 于 2015 年提出，这是一个深度神经网络训练的技巧，它可以在一定程度缓解了深层网络中“梯度弥散”的问题，从而使得训练深层网络模型更加容易和稳定。

具体的训练和测试流程如图所示。

Input: Network N with trainable parameters Θ; subset of activations $\{x^{(k)}\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network

2: **for** $k = 1 \ldots K$ **do**

3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)

4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead

5: **end for**

6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$

7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters

8: **for** $k = 1 \ldots K$ **do**

9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.

10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$

$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$

12: **end for**

3.2 Instance Normalization

IN 即实例正则化。是对一个批次中单个图片进行归一化，而不是像 BN 一样对整个批次的所有图片进行归一化，提取出平均值等。

对于 IN，其具体的公式如下：

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2.$$

3.3 修改方案

在卷积神经网络中，每一层的 feature_map 是高维向量，所以做归一化时也应按照各个维度，比如某一层的输出结果为[batch_size, height, width, channel]四维向量，那么计算的数据量为 height*width*channel。

在修改中，只需要修改 models.py 中_instance_norm()函数中计算均值和方差时调用的 tf.nn.moment()方法，修改原先的计算维度[1,2]为[0,1,2]，如下。

```
tf.nn.moments(net, [0,1,2], keep_dims=True)
```

3.4 实验结果

我们记录了 Batch-Normalization 版本中训练集每一步的各种损失函数数值，经过每 100 步取平均值平滑后，四种 Loss 曲线如图 3。

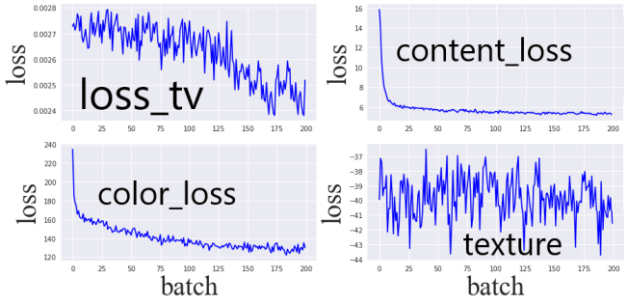

图 3. Batch-Normalization 四种损失函数下降曲线

可见，除了 Texture 损失以外，其余均呈现下降趋势，其中 Total-variation 损失较小，基本不变。关于 Texture 损失，我认为这是判别器和生成器对抗的结果，导致判别器的准确率最终在 0.5 附近波动，对应的交叉熵也在某数值附近波动，Batch-Normlization 的训练姐和测试集 Total_Loss 如图 4.

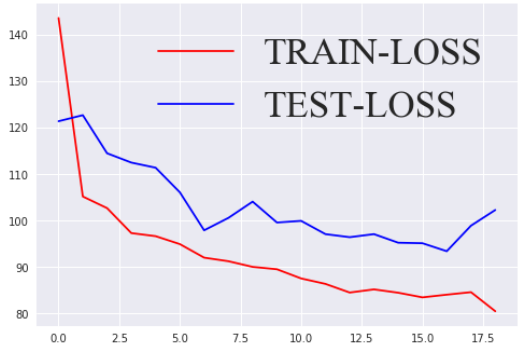

图 4. Batch-Normalization 训练集与测试集 Loss 之和曲线

图 5 分别是迭代次数为 1 和 19000 次时，生成器在测试集上的真实图片效果，可见生成器产生了较好的效果。

图 5. Batch-Normalization 真实图片效果

四. DENSENET 生成器

原论文的生成器采用的是经典的 Resnet，残差结构提高了收敛速度，缓解了梯度爆炸、消失等问题，而 Densenet 作为 2017 年 CVPR 最佳论文，我们认为

其结构与 Resnet 有相通之处（层与层连接更加紧密），采用它可能能够帮助提高生成效果。

4.1 Densenet 网络结构简述

用一幅图简要说明 Densenet 网络结构，如图 6，Densenet 网络采用第一层连接第二层、第三层、第四层……第二层连接第三层、第四层…….以此类推的结构，论文提到这样的方式能够产生正则化效果以及和 Resnet 类似的效果。同时，网络结构中存在多个卷积层相连构成的所谓 Dense-Block，Block 之间通过 Transition 层连接，结构比较清晰，易于上手，但是论文也提到由于多个卷积层连在一起作为一个块这样的操作会导致显存消耗十分巨大，尽管经过调研后来提出了 Efficient-Densenet 解决了显存消耗问题，但是在实践中，通过设置合适的规模参数(batch_size 等)，我们并没有遇到显存不够用的问题，故没有在深入研究后续效率改进相关的工作。

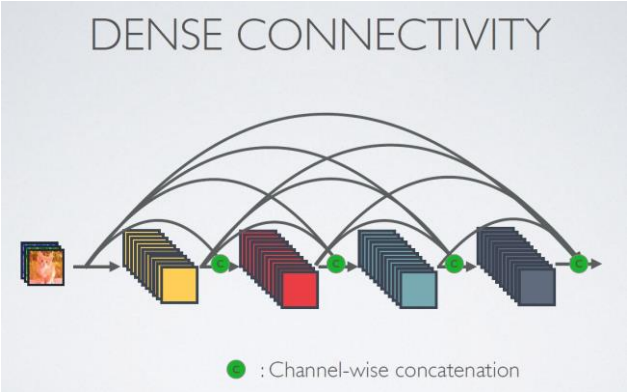

图 6. Densenet 网络结构

4.2 代码分析

我们利用 Tensorflow 框架，参考原论文的 Resnet 的方式，实现了一个简单的 Densenet，由于原论文 Densenet 是作为一个分类器而存在的，与本文生成器的任务不相符合，因此我删除了论文结构中的 Softmax 层，换成与输入图像对应的普通卷积层，将输出的 feature_map 经过 tanh 函数激活后作为生成图像。网络总体结构在附录中给出，下面我们介绍一下最为关键的 Dense-Block 的实现方式，如下，我们创建了 12 个普通卷积层（ReLu 激活），将他们连接在一起从而构成一个 Block，共有 3 个 Block。

```
def dense_block(x, name, layers=12):
    for i in range(layers):
        W = weight_variable([3,3,16,16], name=name+"_W"+str(i))
        b = bias_variable([16], name=name+"_b{0}".format(i))
        x = tf.nn.relu(conv2d(x, W, 1) + b)
        if i == 0:
            x_con = x
        else:
            x_con = tf.concat([x_con, x], 3)

    return x_con
```

4.3 实验效果

由于参数需要调整，尽管我们尝试了多次，耗费了大量人力和资源，但仍未找到超越原 Resnet 效果的超参数，然而下降曲线和真实测试图片都反映了 Densenet 网络的正确性，最终的 SSIM 指标为 0.90，PSNR 指标 18，与论文的 0.93 存在差距，需要指出的是，这里用到的归一化层仍然是 Instance-Normalization，测试集和训练集的 Loss 下降曲线如图 7。

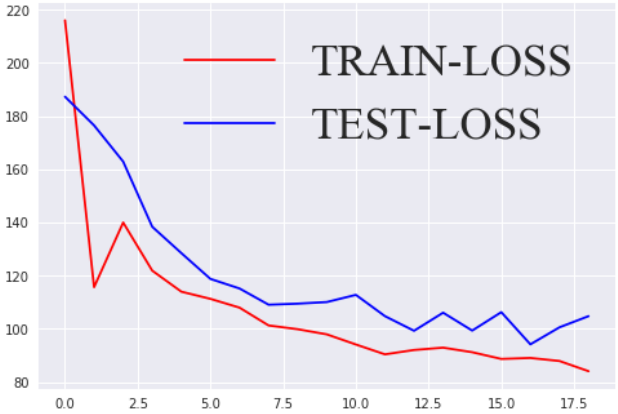

图 7. Densenet 网络损失函数曲线

判别器在测试集和训练集上的变化如图 8，其中蓝色为测试集，红色为训练集，它们最终在 0.6 附近波动，这反映其实网络训练得还不够完美，通过学习率等设置，应该还能有所提升。

在真实图片测试上，Densenet 也取得了可观的效果，图 9 分别是迭代次数为 1000 和 19000 次时的生成对比图。

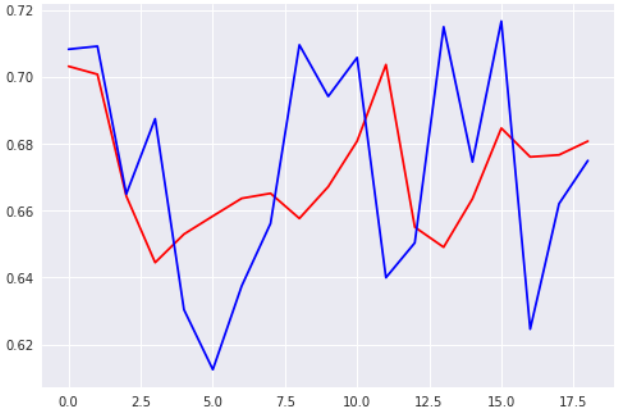

图 8. Densenet 网络判别器准确率曲线

图 9. Densenet 网络迭代次数为 1000（上）和 19000（下）次的图片测试对比

五. CYCLE-LOSS 与 UNPAIRED 训练

在调研 GAN 网络相关知识时，最吸引我们的便是风格迁移的应用，深入研究后，发现它们大多采用一种称为 Unpaired 的训练方式，即生成器从 A 域到 B 域可以不要求两边的图片必须匹配，而 Paired 则要求必须存在两张能够匹配的图片分别位于 A 域和 B 域。举一个例子，在 DPED 中，作者采用不同的手机和单反拍摄相同位置的图片（甚至进行了图像对齐操作），这里存在 Blackberry 域、iPhone 域、Sony 域以及 DSLR 域，在四个域中存在同一场景的图片，它们是一一匹配的。

类似有监督学习和无监督学习的关系，Paired 方式对人力、物理要求较高且应用存在局限性（例如如果要进行 Paired 风格迁移则需要花费更多的力气在人工迁移上……），而 Unpaired 则给与我们无限的想象空间，本文参考 Cycle-GAN，修改原论文的结构为 Unpaired，并加入特有的 Cycle-Loss。

6.1 Cycle-GAN 与 Cycle-LOSS

在 Cycle-GAN 中存在两个判别器和两个生成器，它们的结构如图 10 所示。

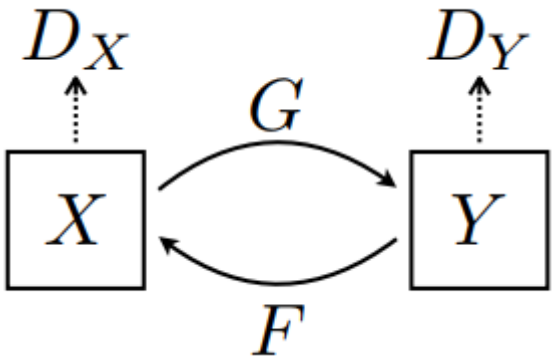

图 11. Cycle-GAN 网络结构

在 Cycle-GAN 中，其目的是将 X 域的对象转化为带有 Y 风格的对象，在我们的场景中，X 域即手机拍摄的图片，Y 域即单反拍摄的图片，这里应当抛弃一一匹配的概念，然后有**两个不同的生成器** G 和 F，分别用于将 X 转化为 Y 和 Y 转化为 X，以及**两个不同的判别器** Dx 和 Dy 分别用于判断真假 X 和真假 Y。

与传统 GAN 相同，为了保证生成器的效果，我们采用生成对抗的方式，以判别器结果的交叉熵作为损失函数去训练从而提高生成器的能力，由于存在两个生成器，因此有两个损失函数，其中 X 到 Y 方向的损失函数如下，反方向类同。

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

但是这样会产生一个问题，如果 X 域生成器妄图欺骗判别器，它会直接生成一个 Y 域的图像，Y 域生成器同理，那么这样**虽然降低了损失函数，但是与实际的训练目标（图像增强或风格迁移等）相违背**。

为此，Cycle-GAN 提出了 Cycle-loss，首先来看损失函数表达式，如下。

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

表达式的含义是 X 域真实图片与 Y 域生成器在 X 域到 Y 域生成图片上的输出应当近似，对称同理。这里差异度量采用的是 L1 范数，即绝对值求和，关于为什么采用 L1 范数，作者并未详细解释，但是提到也测试了其它测度，如 L2 范数等，但是效果并没有明显提升，我们姑且将其视为类似超参数的角色。

关于 Cycle-Loss，我们认为可以这么理解，控制了某域的变化范围，**生成器的作用应当尽量保证可逆，即可以用反方向生成器逆变换变回来**，一言概括即“**绑定**”，这样能够解决上面提到的对称交叉熵之和作为损失函数的问题。

6.2 Tensorflow 实现

由于原论文采用是 Paired 的方式，并且只有单个生成器和判别器，因此修改起来还是大费周折的，下面描述我们修改的过程。

首先需要增加生成器和判别器的数量，这里增加的仍然是 Resnet 和普通卷积神经网络，也即现在网络结构中存在两个生成器：Generator_A2B(resnet), Generator_B2A(resnet) 和两个判别器：Discriminator_B(CNN)、Discriminator_A(CNN)。

其中，两个生成器会产生**四种**结果，X 到 Y 的生成结果（记为 X->Y）、Y 到 X 的生成结果（记为 Y->X），（X->Y）到 X 的生成结果，(Y->X)到 Y 的生成结果，如下。

```
enhanced = models.generator_A2B(phone_image, None)
r_enhanced = models.generator_B2A(dslr_image, None)
r_enhanced_enhanced = models.generator_B2A(enhanced, True)
enhanced_r_enhanced = models.generator_A2B(r_enhanced, True)
```

然后计算判别器的交叉熵，是两个方向对称的，其和作为判别器损失函数并入到原论文的 loss_texture 中，如下。

```
loss_discrim = -tf.reduce_sum(discrim_target * \
               tf.log(tf.clip_by_value(discrim_predictions, 1e-10, 1.0)))
r_loss_discrim = -tf.reduce_sum(r_discrim_target * \
                 tf.log(tf.clip_by_value(r_discrim_predictions, 1e-10, 1.0)))
loss_disc = loss_discrim + r_loss_discrim
```

按照论文的 L1 范数之和，我们描述了 Cycle-Loss，如下。

```
cycle_loss = tf.reduce_sum(tf.abs(r_enhanced_enhanced - phone_image)) \
           + tf.reduce_sum(tf.abs(enhanced_r_enhanced - dslr_image))
```

必须注意的是，在计算生成器结果的时候，我们调用了四次生成器，如果采用默认方式，Tensorflow 会构建出四个生成器网络流图，从而训练出四个（X->Y->X 和 Y->X 应该是同一个生成器而非两个），为了解决该问题，我们利用 Tensorflow 的变量名字空间机制，在第二次使用相同名字的变量时应当查看命名空间中已存在的变量并复用它，而非重建一个新的。具体来说，我们在生成器网络入口送入 isuse 参数，决定这里的参数是否复用，然后在创建命名域的时候设置 reuse 参数为 None（第一次）或 True（第二次及以后），如图。

```
def generator_A2B(input_image, isuse):
    with tf.variable_scope("generator_1", reuse=isuse):
```

还有一点，原论文的 Instance_norm 没有采用命名方式，这里需要手动添加，以及原论文中创建卷积核参数和偏置参数时应当改为 get_variable 方式创建变量从而使用“**复用机制**”。

6.3 实验效果

在实验中，我们发现原论文的 color_loss 很难下降，仔细分析后我们观察到 color_loss 与 cycle_loss 的形式很是相似，如果忽略高斯模糊、L2 范数的话，但是 color_loss 是单向的，故决定删除 color_loss。

另外，超参数设置也很重要，我们主要调节的参数是学习率和 cycle_loss 的权重。第一，我们观察到虽然 loss 有所下降，但是收敛速度过慢，导致我们耗费了大量时间等待无用训练，在尝试提高学习率（源代码是 5e-4）为 1e-3 后，效果比较好，当然学习率也不能设置过高，否则在大约 10000 次迭代后，loss 偏向抖动，无法适应后期的精度。第二，cycle_loss 往往很大（初期 10W+），我们开始想通过增大 cycle_loss 权重，让其下降更快，然而导致无法收敛（可能是导致其他 loss 效果降低），后来采取较小的权重(0.01)，发现效果还不错，loss 稳步降低。

最终，在测试集和训练集上，我们的 loss 损失曲线如图 12，而关键的 Cycle-Loss 则在图 13 中。

在真实图片上，图 14 展示了迭代次数为 1000 和 8000 次时的对比效果图，发现提升效果不错，但是仍有上升空间（由于在手动 Grid_Search 调节超参数上耗费了大量时间，我们不得不将迭代次数调节为 8000，如果有充足的训练时间，效果应当会更好）。

图 12. Unpaired 训练方式损失函数曲线

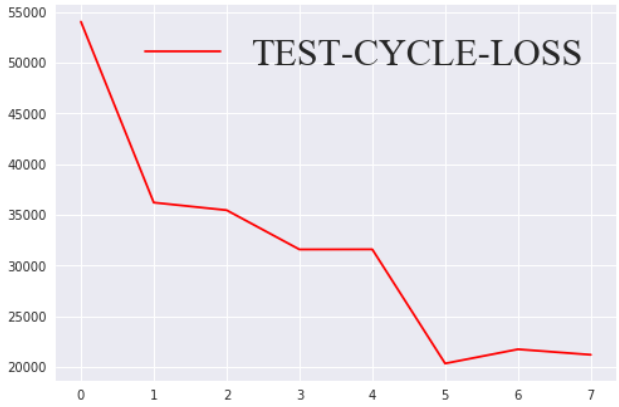

图 13. 测试集 Cycle-Loss 曲线

图 14. Unpaired+Cycle-loss 网络迭代次数为 1000（上）和 8000（下）次的图片测试对比

六. 思考与总结

本文基于 DPED 论文，尝试了多种改进方案去完成图像增强任务，包含：原论文复现（Instance-Norm）、Batch-Normalization 层、Cycle-Loss 损失函数、Densenet 生成器以及 Unpaired 训练方式。尽管由于参数调节等问题，最终效果并未超过 DPED 论文，但是在实验的过程中，我们拓宽了视野，学习到了更多与多媒体紧密相连的知识。

关于两种归一化层的选择，原论文采用 Instance-Normalization，而我们尝试修改为 Batch-Normalization，效果降低。我们认为这不仅是结构的改变不适应原参数导致的，事实上，BN 在图像分类任务上有突出的效果。因为图像分类不需要保留图像的对比度信息，利用图像的结构信息就可以完成分类，所以，将图像都通过 BN 进行归一化，反而降低了训练难度，甚至一些不明显的结构，在 BN 后也会被凸显出来（对比度被拉开了），而对于照片风格转移，使用 IN 效果则更为明显，因为 IN 是比 BN 更直接的在图像像素上对单幅图像进行的归一化操作。而 BN 会忽略图像像素（或者特征）之间的绝对差异（因为均值归零，方差归一）。所以 BN 在效果上不如 IN 也情有可原。

关于生成器的选择，我们修改原来的 Resnet 为 Densenet，通过网络结构对比，Densenet 和 Resnet 有很多相似之处，residual-block 和 dense-block 均采用级联多层作为一个新的网络单元，只不过 dense-block 中各层连接的更为紧密，我们认为 Densenet 不应当弱于 Resnet，实验中的效果低是由于参数设置以及网络结构设置导致的，我们参照 Densenet 论文结构实现的生成器，但是该论文是针对图像分类任务的，其结构不一定适应生成式任务。

有趣的 Cycle-GAN 一经提出便受到广泛关注，类似有监督和无监督的关系，Cycle-GAN 使用了 unpaired 的训练方式，对于两个域的对象不再需要一一匹配，极大地扩展了应用范围。在实验中，我们尝试将图像增强修改为 unpaired 的方式并加入 cycle-loss 绑定两个生成器从而控制生成范围，这一步着实花费了很多精力，Tensorflow 命名空间机制的学习、变量复用帮助我们不需要重构论文的生成器结构（因为原文的 resnet 接口只返回生成结果，没有提供 train 和 predict 接口，这导致多次使用 resnet 会在网络图中构建多个不同的 resnet，而我们只需要一个或两个）。然后是 loss 权值的调节，在实验中，我们发现设置不同的 loss 会导致不同的结果，有些参数甚至会导致 texture_loss 保持恒定，训练失败，而有些参数又会导致训练无法收敛，最后经过慢慢尝试，我们提高学习率(1e-3)，降低 cycle_loss 权重(0.01)才得到一个比较不错的结果。

总的来说，通过本次实验，我们学到了很多知识，抛开较枯燥的数学知识以外（当然，我们也投入了大量时间钻研这些数学公式），图像增强、风格迁移本身十分有趣，它们与多媒体技术的联系很紧密，借助于此，我们提高了对多媒体技术的深刻认知。

参考文献

[1] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks", Computer Vision (ICCV) 2017 IEEE International Conference on, pp. 2242-2251, 2017

[2] G. Huang, Z. Liu, K. Q. Weinberger, and L. Maaten. Densely connected convolutional networks. In CVPR, 2017

[3] Geoff Pleiss, Danlu Chen, Gao Huang, Tongcheng Li, Laurens van der Maaten, and Kilian Q Weinberger. Memory-efficient implementation of densenets. arXiv preprint arXiv:1707.06990, 2017 9

[4] A. Ignatov, N. Kobyshev, K. Vanhoey, R. Timofte, L. Van Gool, "DSLR-quality photos on mobile devices with deep convolutional networks", Proceedings of IEEE International Conference on Computer Vision (ICCV), Oct 2017. R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, “High-speed digital-to-RF converter,” U.S. Patent 5 668 842, Sep. 16, 1997

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele.. The cityscapes dataset for semantic urban scene understanding. In CVPR, 2016

[6] Victor Lempitsky Dmitry Ulyanov Andrea Vedaldi. “Instance Normalization: The Missing Ingredient for Fast Stylization”. In: arXiv preprint arXiv:1607.08022 (2016)

[7] Rashad Al-Jawfi. Handwriting arabic character recognition lenet using neural network. Int. Arab J. Inf. Technol., 6(3):304–309, 2009

[8] S. Targ, D. Almeida, K. Lyman, Resnet in resnet: Generalizing residual architectures, 2016, [online] Available

[9] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In NIPS, pages 1486–1494, 2015

[10] 统计机器学习，李航，清华大学出版社，2017