基于自编码器和分类器的灰度图像上色

自 64　　赵文亮　　2016011452

2019 年 1 月 20 日

1　引言

灰度图像上色是计算机视觉中的一个经典的问题，并具有广泛的应用范围（例如对历史照片上色、对彩色图像重上色等）。一般来说，想从一张灰度图像恢复出原始的彩色图像是一个病态问题，无法得到唯一解。尤其是在没有用户输入的情况下，完成这一任务更加困难。本文使用深度学习的方法，通过在 ImageNet 中的 8000 张图片的数据集上训练一个基于自编码器和分类器的神经网络，实现了一个灰度图像上色器，取得了较好的效果。本文后续章节的内容如下：第 2 节中首先介绍了灰度图像上色问题的研究背景，第 3 节介绍了模型的构成，第 4 节展示了灰度图像上色的结果，最后对实验进行总结。

2　问题背景

灰度图像上色是一个十分困难的问题，仅仅靠输入的灰度图几乎不可能实现。通常的解决方法大致可以分为两种：用户手动标记或使用先验知识。第一种方法要求用户在图片上手动分割区域或手动用线条标记颜色。例如，[1] 中提出了一种基于优化方法的灰度图上色。该方法基于这样一个假设：相邻像素的灰度如果相似，那么颜色也应该相似。该方法在 YUV 颜色空间进行，将灰度图作为 Y 通道，再利用用户在灰度图上标记的不同颜色线条求解优化问题得到 UV 通道，最终得到彩色图像；第二种方法需要一定的先验知识，例如 [2] 需要用户提供一张引导图像，最终将引导图像的颜色迁移到原灰度图上。然而这种方法仍然需要用户输入，没有做到全自动的图像上色。本文中实现的算法基于 [3] 提出的模型，具体的模型构成再下一节中会给出。对于灰度图上色问题，使用机器学习方法的一个好处是数据集充分。任何一张彩色图像都可以作为训练的样本。虽然在训练阶段需要的时间较长，但是一旦模型训练结束，在实际的应用中的处理速度很快（<50ms）。

3　模型构成

本文中的算法在 CIE L*a*b* 颜色空间实现。其中 L* 表示亮度通道，a* 和 b* 为颜色通道（绿-红和蓝-黄）。给定一张大小为 $H \times W$ 的灰度图片，我们可以认为灰度图的亮度与实际的亮度相同，即可得到亮度通道 $X_L \in \mathbb{R}^{H \times W \times 1}$。算法的目的是通过亮度通道 X_L 得到三个通道的估计值 $\tilde{\boldsymbol{X}} \in \mathbb{R}^{H \times W \times 3}$，所以我们只需要得到从 L* 通道到 a* 和 b* 通道的映射关系：

$$\mathcal{F} : X_L \to (\tilde{X}_a, \tilde{X}_b) \tag{1}$$

其中 $(\tilde{X}_a, \tilde{X}_b)$ 表示模型预测得到的颜色通道的值。再加上已知的亮度通道信息，并反变换到 RGB 颜色空间，即可得到所需的彩色图像。所以，网络的输入是一张灰度图像（$H \times W \times 1$），输出是 a*b* 通道（$H \times W \times 2$）。本文中的模型使用了类似于自编码器的架构 [4]。然而单纯使用灰度图表面的信息进行训练并

不能得到很好的结果。事实上，图片中的物体颜色很大程度上取决于物体的种类，充分利用这个信息可能会得到更高的结果。Inception-ResNet[5] 是目前分类任务中表现最突出的网络之一，本文的模型中使用预训练的 Inception-ResNet-v2 来得到图片的类别特征，并将该特征作为另一个输入融合在网络中。

3.1 自编码器

传统自编码器的主体结构如图 1 所示。输入 X 经过一系列隐藏层编码得到 z，再通过一系列隐藏层解码得到 X'。对于图像处理问题，图中的隐藏层通常使用卷积层实现，本文中的模型也是如此。

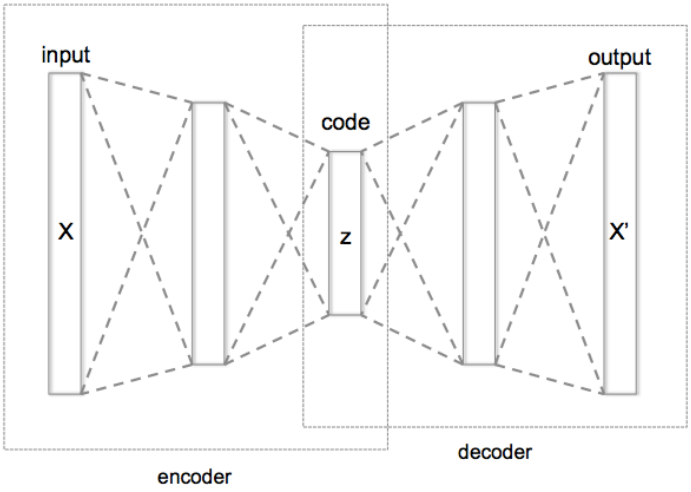


图 1: 自编码器结构

由于本文中的问题与传统的自编码器处理的问题目的不同，网络的输入输出也有所不同。本文中模型的输入为灰度图，输出为预测的 a*b* 两个通道的图片。

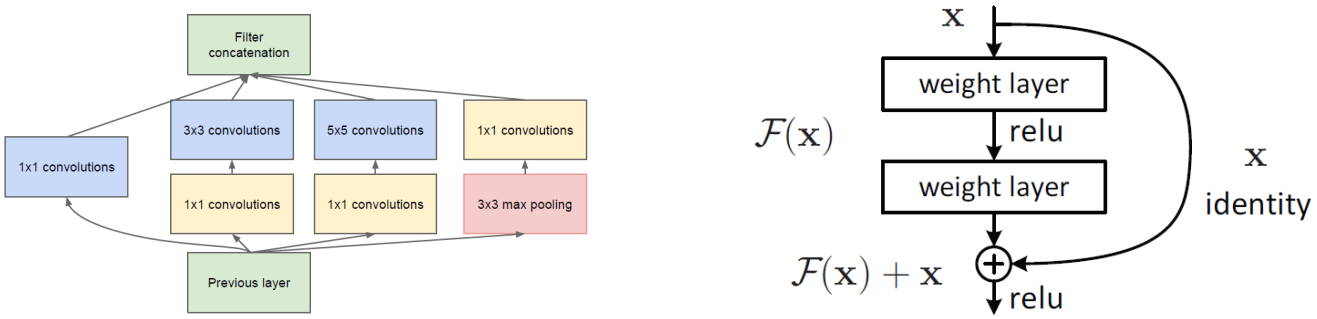
3.2 分类器

本文中使用的分类器是 Inception-ResNet，这是一种将 Inception [6] 和 ResNet[7] 的核心思想结合起来构建的网络。Inception 模块的结构如图 2a 所示。该模块使用卷积核大小不同的卷积层并联，可以很好地提取不同尺度的特征；在 3x3 和 5x5 的卷积层前串联 1x1 的卷积层，可以有效减少网络参数的数目从而加快训练速度。ResNet 模块的结构如图 2b 所示。设模块输出 $\mathcal{H}(x) = \mathcal{F}(x) + x$，则 $\mathcal{F}(x) = \mathcal{H}(x) - x$。[7] 假设，对残差 $\mathcal{F}$ 的学习会比直接对 $\mathcal{H}$ 的学习容易。分别以上述两种结构为基础的 GoogLeNet 和 ResNet 在图像分类、图像识别等任务中都取得了很好的成绩。

[5] 则尝试将 Inception 和 ResNet 结合起来，得到了图 3 所示的模块。课件该模块将 ResNet 模块中的卷积层替换成了 Inception 模块中的结构，实验表明，Inception-ResNet 确实能够发挥二者的优点，在 ILSVRC 数据集上表现出了更好的效果。

3.3 模型结构

本文中使用的模型结构如图 4 所示。主要分为编码、融合、解码三个部分。



(a) Inception 模块 (b) ResNet 模块

图 2: Inception 与 ResNet 基本模块

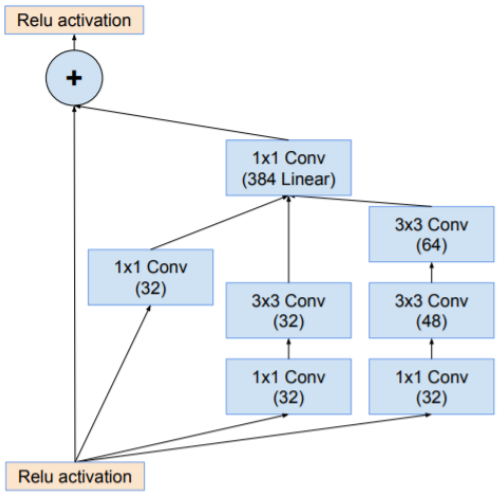


图 3: Inception-ResNet 模块

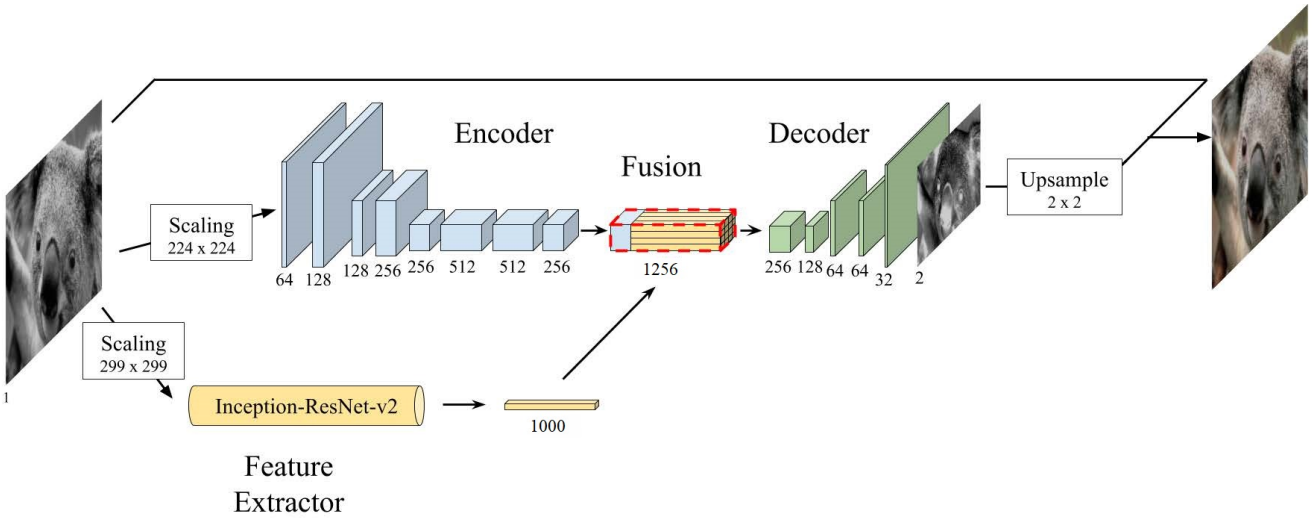


图 4: 模型结构

编码 编码器的输入为 $H \times W \times 1$ 的灰度图，输出为 $H/8 \times W/8 \times 256$ 的图片。此处 $H = W = 224$，即 $H/8 = W/8 = 28$。训练时，将所有训练集的图片缩放到 $\max H, W = 224$，再用白色补成 224×224 的正方形。编码器中的下采样（第 1、3、5 层）通过步长为 2 的卷积层而非池化层实现，下采样可以有效减少计算量。在每一次下采样后，我使用了 Batch Normalization 进行归一化。

融合 将灰度图缩放到 299×299 大小（保持长宽比，用白色补成正方形）后，输入到预训练的 Inception-ResNet-v2 模型，可以得到一个 1000×1 的特征输出。将此输出在宽度和高度方向重复 28*28 次可以得到一个 $28 \times 28 \times 1000$ 的图像，接着将其与编码器的输出在深度方向连接即可完成融合，这个过程在图 4 中有直观的显示。

解码 将融合后的结果经过五层卷积和两次上采样，可以得到 $H \times W \times 2$ 的输出图片，这就是我们想要的 a*b* 通道。同样在每次上采样后增加了 Batch Normalization 层。

4 实验与分析

4.1 数据预处理

本文使用 ImageNet 中的 8000 张图片进行训练。在训练之前，需要进行数据的预处理。首先将图片缩放并补白边得到 299×299 的图片，如图 5 所示（为了便于观察，为图片绘制了边框）。

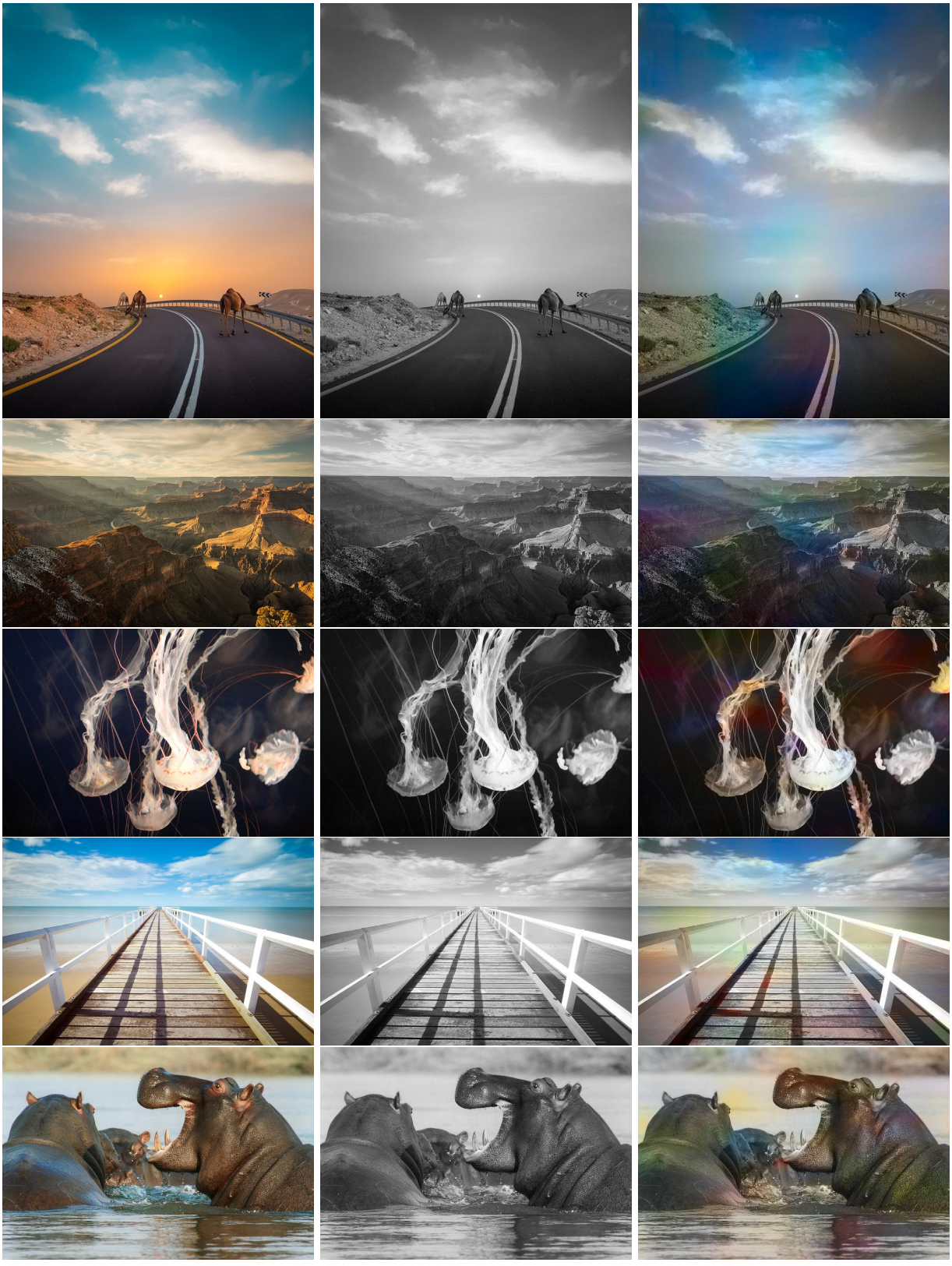
(a) 原始图片 (b) 缩放后图片

图 5: 图片预处理示例

考虑到对于任何一张图片，Inception-ResNet-v2 的输出是固定的，所以事先将输出计算好保存下来可以节省训练中的时间。另一方面，由于数据量过大，一次性读取到内存中是不现实的，需要逐块从文件中读取。TFRecords 是解决该问题的一种很好的方式。每一条 TFRecord 中包含的内容如下：

1. 灰度图（$299 \times 299 \times 1$）
2. a*b* 图像（$299 \times 299 \times 2$）
3. Inception-ResNet-v2 的输出（1000×1）
4. 图像文件名

对每张图像计算得到上述信息后，保存在一个 TFRedords 文件中。训练时可以通过 tf.data.TFRecordDataset 来读取。



(a) (b) (c)

图 6: 实验结果：c：彩色图片；b：灰度图片；c：上色后图片

4.2 模型训练

我使用 Keras[8] 对本文中的模型进行实现，训练在 *NVIDIA GeForce GTX Titan Xp* 上完成。训练时随着轮数增加递减学习率：初始学习率为 1e-3，在 100、150、200、300 轮次时将学习率除以 10。使用 Adam 优化器、MSE 损失函数进行训练。

4.3 实验结果

实验结果表明，模型在大约 400 个 epoch 时已经收敛，此时的损失函数约为 30（初始为 220+）。实验中发现。直接使用训练好的模型对灰度图片上色的效果饱和度略低，所以我将饱和度乘以一个系数 1.2 作为调整。上色结果如图 6 所示。需要指出的是，想让上色结果与原来的彩色图像完全相同是不可能的，我们能做的只是让输出的图片更加接近人的视觉感受。从图 6 中可以看出，图片上色的效果总体来说较好。例如第 1、2、5 行成功将天空区域填充了蓝色；3、5 行水母和河马的颜色也比较真实。不足之处在于，上色后图片中某些区域存在一些颜色不均匀的现象，这些现象可能通过一些后续的处理手段解决（例如结合 [1] 中的算法）。

5 总结

本次实验中，我实现了灰度图片上色的算法，并取得了较好的效果。选择这个问题的原因是我觉得这个问题比较有趣而且富有挑战性。选择使用深度学习的方式来实现有多方面的原因：首先老师在上课时经常提到使用深度学习来完成数字图像处理的例子；其次我想通过这个大作业锻炼一下训练神经网络的能力；最后针对本问题而言，我希望实现一种全自动的算法，不需要用户的任何输入。

在完成实验的过程中，我有了多方面的收获。在文献调研中，我了解了许多深度学习中的概念，并阅读了大量的优秀论文，学习了一些有创新性的网络架构（GoogLeNet、ResNet 等）。在编写代码时，我学会了使用 Tensorflow 和 Keras 搭建神经网络，也深刻体会到了 Batch Normalization 的重要性（实验中不加归一化时完全不收敛）。

实验中一个比较困难的点是数据的预处理。一开始我没有找到有效的读取 TFRecord 的方法，浪费了大量的时间。后来发现了 TFRecordDataset 这个高层的 API，它能够从 tfrecords 中逐块读取到内存中，并有打乱、重复取样等功能，并可以与 Keras 的 API 完美结合。查阅相关文档后，我顺利地完成了数据的预处理和读取。

本学期的数图就此结束了，回顾这学期的课程、作业、考试，虽然难度较大，但是在认真完成的过程中确实能够收获到不少知识。感谢老师的悉心教导和助教的辛勤付出！

参考文献

[1] A. Levin, D. Lischinski, and Y. Weiss, “Colorization using optimization,” in *ACM transactions on graphics (tog)*, vol. 23, pp. 689–694, ACM, 2004.

[2] T. Welsh, M. Ashikhmin, and K. Mueller, “Transferring color to greyscale images,” in *ACM Transactions on Graphics (TOG)*, vol. 21, pp. 277–280, ACM, 2002.

[3] F. Baldassarre, D. G. Morín, and L. Rodés-Guirao, “Deep koalarization: Image colorization using cnns and inception-resnet-v2,” *arXiv preprint arXiv:1712.03400*, 2017.

[4] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, 2014.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI*, vol. 4, p. 12, 2017.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[8] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.