静止图像压缩编码——传统方法和深度学习方法对比

仝顽杰

（学号：2020110161 学院：信息与通信工程学院 email：twj@bupt.edu.cn）

摘 要：传统的静止图像压缩编码主要包括 JPEG 和 JPEG2000，基于深度学习的图像压缩编码近年来发展活跃，而且，基于神经网络的更通用的压缩框架可能能够更快地适应那些不断变化的任务和环境。本文主要针对 JPEG、JPEG2000 以及自动编码器这三种静止图像压缩编码方案进行了原理介绍、流程分析、编程实现和性能对比。从性能上看，基于深度学习的自动编码器在图像压缩方面具有很大的潜力，而且，自动编码器也符合端到端通信的需求，这方面的工作将会对之后的研究打造一定的基础。另外，所有的实现都提供相应的代码。[1]

关键词：静止图像压缩编码；JPEG；JPEG2000；深度学习；自动编码器

1 引言

传统通信技术已经发展到了 5G，在第一、二代移动通信技术中，主要的业务是语音，然而自从 3G 以后，通信业务推广为包含语音、数据与图像的多媒体业务。随着 4G 的出现，移动互联网加速形成，如今，短视频业务已经随处可见。据爱立信最新移动性报告数据，移动视频流量每年增长约 50%，到 2023 年占据全部移动数据流量的 75%。

伴随着越来越多的图像与视频业务，通信资源变得越来越匮乏，我们知道，图像或者视频如果不经压缩直接传输的话，将会占用极大的带宽资源，早在上世纪 90 年代，联合图像专家组（Joint Photographic Experts Group，即 JPEG）就推出了一种静止图像压缩编码标准，简称 JPEG。自从它在 1992 推出以来，如今已经成为世界上使用最广泛的图像压缩标准以及使用最广泛的数字图像格式，截至 2015 年每天产生数十亿张 JPEG 图像。JPEG2000 是有联合图像专家组于 2000 年左右提出的新的图像压缩编码标准，旨在取代基于原始离散余弦变换的 JPEG 标准。但是由于 JPEG2000 在很多 Web 浏览器中都不支持，所以其在 Internet 上用得较少。需要注意的是，本文仅关注静止图像的有损压缩。

无论是 JPEG 还是 JPEG2000，其本质都是对图像进行了特定的变换（离散余弦变换或者离散小波变换），之所以这些变换能起作用，是因为人眼对低频图像信号的变换相比高频部分更加敏感，因此可以采取保持低频的质量，压缩高频的方法来对图像进行一定程度的压缩。然而这些变换都可以被理解为一种人工干预的变换，采用的是事先设计好的各种滤波器，这样的做法是有效的，但却不一定是最优的选择。

近年来，深度学习技术成为一种强大的方法，可以从数据中自动学习特征表示。在 2012 年，Krizhevsky 等人提出了一种称为 AlexNet 的深度卷积神经网络（DCNN），该技术在大规模视觉识别挑战赛（ILSVRC）中获得了创纪录的图像分类精度。从那时起，计算机视觉大多数方面的研究重点一直集中在深度学习方法上。AlexNet 网络的成功，除了庞大的数据集以及高性能的计算能力以外，良好的特征提取网络在目标检测中也至关重要。这个特征提取网络，既做到了特征的自动提取，又具有非常好的性能。尝试将深度学习的方法应用到通信领域，许多端到端的通信架构被研究者提出。从本质上看，传统静止图像压缩技术中的变换操作，可以和深度学习网络中的特征提取网络进行对比，于是，能否设计一种更加高效的网络来实现图像的特征变换，进而实现压缩呢？答案是肯定的，自动编码器就是一种非常有效的端到端图像压缩框架。

[1] https://github.com/Tong-wj/JPEG-JPEG2000-autoencoder.git

本文剩下的部分安排如下：第 2 部分从原理和框架上介绍了传统的图像压缩标准 JPEG 和 JPEG2000，第 3 部分介绍了一种基于深度学习的自动编码器实现的图像压缩编码，第 4 部分介绍了 JPEG、JPEG2000 以及自动编码器的编程实现，第 5 部分对传统的静止图像压缩编码和基于深度学习的方法进行了定量对比，最后，第 6 部分给出总结。

2　传统图像压缩标准 JPEG 和 JPEG2000 的基本原理

2.1 JPEG 基本原理

JPEG 使用基于离散余弦变换（DCT）的有损压缩形式。该数学运算将视频源的每个帧/场从空间（2D）域转换为频域（也称为变换域）。宽松地基于人类心理视觉系统的感知模型会丢弃高频信息，即强度和色彩的急剧变化。在变换域中，减少信息的过程称为量化。简而言之，量化是一种将大数位标度（每个数位出现不同）最佳地减小为较小的方法，而变换域是图像的便捷表示形式，因为高频系数的贡献较小相对于其他系数而言，它们的特征在于具有可压缩性较高的特征值。然后对量化系数进行排序，并将其无损压缩到输出比特流中。JPEG 的几乎所有软件实现都允许用户控制压缩率（以及其他可选参数），从而允许用户权衡图片质量以减小文件大小。在嵌入式应用程序中，将为应用程序预先选择并固定参数。

压缩方法通常是有损的，这意味着某些原始图像信息会丢失并且无法恢复，从而可能影响图像质量。JPEG 标准中定义了一个可选的无损模式。但是，该模式在产品中不受广泛支持。

JPEG 图像由一系列片段组成，每个片段都以一个标记开头，每个标记都以一个 0xFF 字节开头，后跟一个指示其是哪种标记的字节。有些标记仅由这两个字节组成；其他字节后跟两个字节（从高到低），指示随后的特定于标记的有效载荷数据的长度。（长度包括长度的两个字节，但不包括标记的两个字节。）

JPEG 的编码过程包括以下几个步骤

1. 图像中颜色的表示形式转换为 YCbCr，其中，Y 代表亮度分量，Cb 和 Cr 代表色度分量。
2. 色度数据的分辨率通常会降低 2 倍-4 倍，这反映出这样一个事实，即眼睛对精细的色彩细节的敏感度要比对精细的亮度细节的敏感性低。
3. 图像被分成 8×8 像素的块，并且对于每个块，Y，Cb，Cr 数据中的每一个都经过离散余弦变换（DCT）。DCT 产生一个空间频谱意义上的傅里叶变换。
4. 频率分量的幅度被量化。人类视觉对大区域颜色或亮度的细微变化比对高频亮度变化的强度更敏感。因此，高频成分的大小以比低频成分低的精度被存储。编码器的质量设置（质量银子 Q，范围在 0-100 之间）会影响每个频率分量的分辨率降低的程度。如果使用了过低的质量设置，则将高频分量完全丢弃。
5. 使用无损的熵编码（霍夫曼编码、差分编码、算术编码）进一步压缩所有 8×8 块的结果数据。

解码过程会逆转这些步骤，其中量化是不可逆的，因为量化会引入量化噪声。

JPEG 的整体流程如图 1 所示

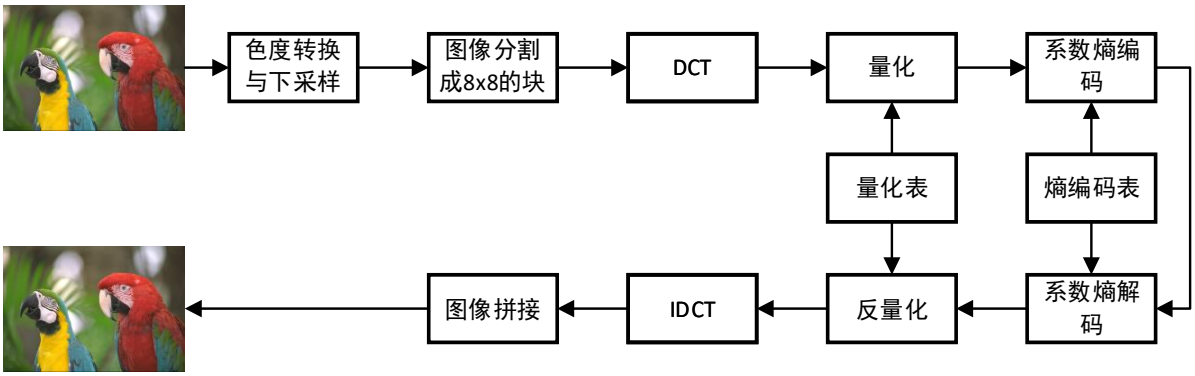


图 1　JPEG 编解码原理框图

需要说明的有：

1. 量化和熵编码均采用相应的量化表或者熵编码表来实现；
2. 为了控制压缩的程度，往往会设置一个质量因子 Q，Q 会影响量化表的数值大小，即量化步长，从而影响压缩程度；
3. 熵编码部分，直流分量采用差分编码，交流分量采用的是游程编码。
4. 虽然框图中给出了编解码整个过程，但是在实际应用的时候，往往只用到一半，即要么只做压缩编码，要么只进行解码，这需要根据场景任务的需要。在这里给出整个编解码过程是为了理解原理。

2.2 JPEG2000 的基本原理

尽管与 JPEG 相比，JPEG 2000 的压缩性能有了适度的提高，但 JPEG 2000 提供的主要优点是码流的显着灵活性。本质上，在使用 JPEG 2000 压缩图像后获得的码流是可伸缩的，这意味着可以用多种方式对其进行解码。例如，通过在任何点处截断码流，都可以以较低的分辨率或信噪比获得图像的表示形式-参见可伸缩压缩。通过以各种方式对代码流进行排序，应用程序可以显着提高性能。但是，由于这种灵活性，JPEG 2000 需要编解码器复杂且计算量大。与 JPEG 相比，另一个区别在于视觉伪像：JPEG 2000 仅产生振铃伪像，表现为图像边缘附近的模糊和振铃，而 JPEG 由于其 8 倍倍率产生振铃伪像和“阻塞”伪像。

1. 和 JPEG 相比，主要有以下改进：
2. 多分辨率表示；
3. 逐像素渐进传输和分辨率精度；
4. 可选择无损或有损压缩；
5. 容错能力；
6. 灵活的文件格式；
7. 高动态范围支持；
8. 边道空间信息。

JPEG2000 编码算法源于 David Taubman 提出的 EBCOT 算法，使用小波变换，采用了两层编码策略，对压缩位流进行分层组织，不仅提高了压缩效率，而且压缩码流具有较大的灵活性．JPEG2000 的编解码系统如图 2 所示，JPEG2000 的编码系统由七个主要模块组成，在 JPEG2000 编码过程中，首先是对原始图像进行离散小波变换，根据用户要求对变换后小波系数进行量化；量化后的小波系数划分为小的数据单元——码块，然后对每个码块进行 独立的嵌入式编码；并将得到的所有码块的嵌入式位流，按照率失真最优原则分层组织，形成不同质量的层。对每一层，按照一定的码流格式打包，输出压缩码流。下面介绍各部分的作用及基本原理。

图像分块与拼接：与 JPEG 不同，JPEG2000 算法并不需要将图像强制分成 8×8 的小块。但为了降低对内存的需求和方便压缩域中可能的分块处理，可以将图像分割成若干互不重叠的矩形块（tile）。分块的大小任意，可以整个图像是一个块，也可以一个像素是一个块。一般分成 $2^6 - 2^{10} \times 2^6 - 2^{10}$（即 64 1024 像素宽）的等大方块，不过边缘部分的块可能小一些，而且不一定是方的．在我们实验的过程中发现如果进行分块处理将会得到更高的运算速度，如果不进行分块处理，需要的内存和运算量均会比分块处理增加不少。

数据偏移和归一化处理：与 JPEG 的 DCT 一样，JEPG2000 中的 DWT 也需要图像的样本数据关于 0 对称分布，对 B 位无符号整数的分量样本值$I(x, y)(0 \le I(x, y) < 2B)$应该先进行中心偏移（又称为直流电平平移 DC level shifting）：

$$I'(x, y) = I(x, y) - 2^{B-1} \tag{1}$$

对于无损压缩，因为采用的是整数小波变换，数据偏移后就可以了．但是对于有损压缩，因为采用的是实数型离散小波变换，所以还需要对偏移后的数据进行归一化处理：

$$I''(x,y)=\frac{I'(x,y)}{2^{-B}} \tag{2}$$

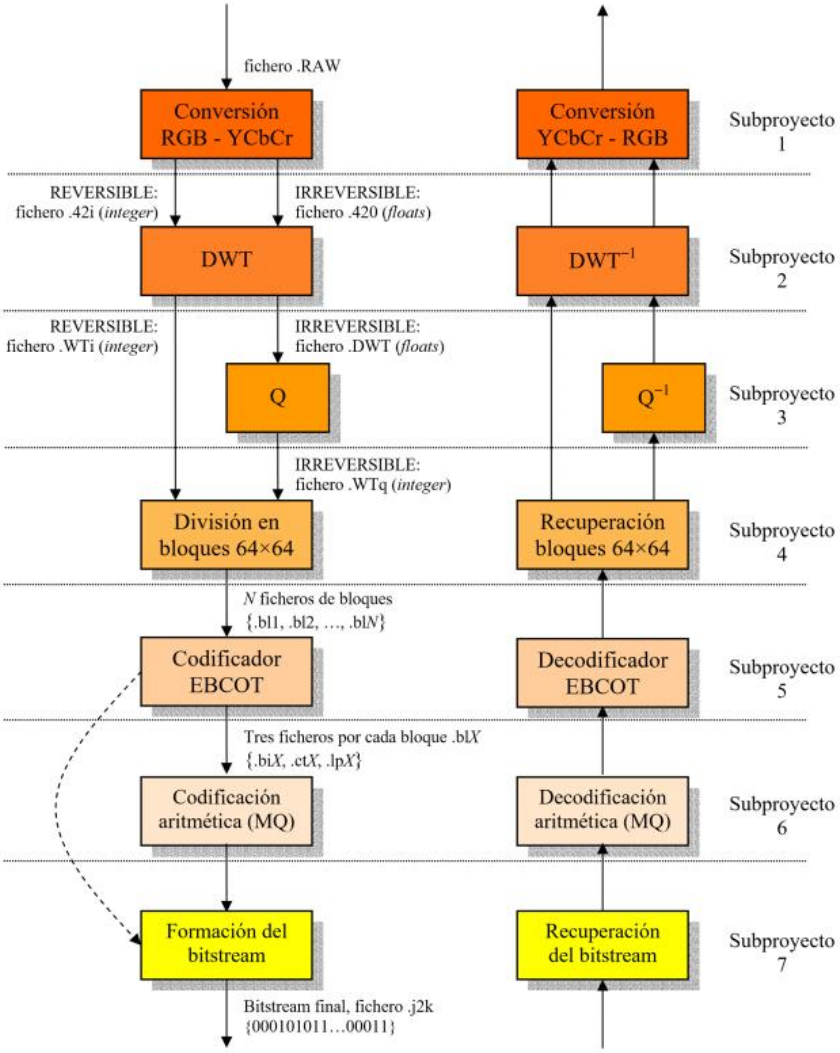


图 2　JPEG2000 编解码原理框图

在解码的后处理中，需要将归一化或者平移后的数据进行还原处理：

$$I'(x,y)=2^{B}I''(x,y) \tag{3}$$

$$I(x,y)=I'(x,y)+2^{B-1} \tag{4}$$

分量变换（可选）：指对具有多个分量的图像先经过某种变换来降低各分量之间的相关性。将传统的 RGB（红绿蓝）色域转换至其他色彩空间。Cr 和 Cb 是差值图像，直方图在零点附近有很高的峰值。分量变换是可选的，只有当图像具有三个或者三个以上的分量，并且参与变换的三个分量具有相同大小和位深。JPEG2000 定义了两种分量变换，一种是不可逆的分量变换（ICT），即实数的运算；另一中是不可逆的分量变换（RCT），即整数的运算．在无损压缩中，只能用 RCT；在有损压缩中，两者都可用。

$$ICT:\begin{pmatrix}Y\\Cr\\Cb\end{pmatrix}=\begin{pmatrix}0.299 & 0.587 & 0.114\\-0.16875 & -0.33162 & 0.5\\0.5 & -0.41869 & -0.08131\end{pmatrix}\begin{pmatrix}R\\G\\B\end{pmatrix} \tag{5}$$

$$iICT:\begin{pmatrix}R\\G\\B\end{pmatrix}=\begin{pmatrix}1 & 0 & 1.402\\1 & -0.34413 & -0.71414\\1 & -1.772 & 0\end{pmatrix}\begin{pmatrix}Y\\Cr\\Cb\end{pmatrix} \tag{6}$$

$$RCT:\begin{pmatrix}Y\\Cr\\Cb\end{pmatrix}=\begin{pmatrix}1 & 0.5 & 0.25\\1 & -1 & 1\\1 & -1 & 0\end{pmatrix}\begin{pmatrix}R\\G\\B\end{pmatrix} \tag{7}$$

$$iRCT:\begin{pmatrix}R\\G\\B\end{pmatrix}=\begin{pmatrix}1 & -0.25 & -0.25\\1 & -0.25 & 0.75\\1 & 0.75 & -0.25\end{pmatrix}\begin{pmatrix}Y\\Cr\\Cb\end{pmatrix} \tag{8}$$

小波变换：图像的二维离散小波分解和重构过程如图 3 所示，分解过程可描述为：首先对图像的每一行进行 1D-DWT，获得原始图像在水平方向上的低频分量 L 和高频分量 H，然后对变换所得数据的每一列进行 1D-DWT，获得原始图像在水平和垂直方向上的低频分量 LL、水平方向上的低频和垂直方向上的高频 LH、水平方向上的高频和垂直方向上的低频 HL 以及水平和垂直方向上的的高频分量 HH．重构过程可描述为：首先对变换结果的每一列进行以为离散小波逆变换，再对变换所得数据的每一行进行一维离散小波逆变换，即可获得重构图像．由上述过程可以看出，图像的小波分解是一个将信号按照低频和有向高频进行分离的过程，分解过程中还可以根据需要对得到的 LL 分量进行进一步的小波分解，直至达到要求。

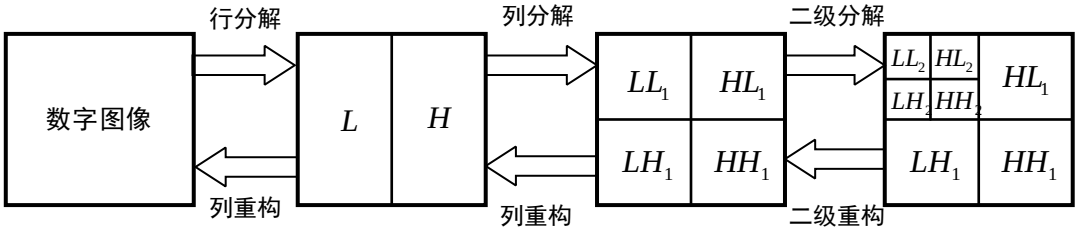


图 3 图像的二维离散小波分解和重构过程

在 JPEG 2000 的核心编码系统中，对有损压缩采用的是基于 Daubechies 9/7 滤波器——提升实现的不可逆 DWT．对无损压缩采用的则是基于 Le Gall 5/3 滤波器——提升实现的可逆 DWT。JPEG 2000 标准支持基于卷积 (convolution-based) 和基于提升 (lifting-based) 两种滤波模式。

量化：对变换后的系数进行量化是有损压缩中的重要环节。因为人眼视觉系统对图像分辨率有一定的界限，通过适当的量化减小变换系数的精度，可以在不影响图像主观质量的前提下，进一步提高图像的压缩比．量化的关键是根据变换后的图像的特征、重构的图像的质量要求等因素设定合理的量化步长．经过小波变换，片（tile）分量数据变换成子带形式，若进行 N 级小波分解，可以得到 $3N+1$ 个子带，每个子带系数都反映了图像不同频率的特征，JPEG2000 标准中采用均匀量化，但不同子带使用不同的量化步长．系数量化的算法为：

$$q_b(u,v)=sign(a_b(u,v))\left|\frac{a_b(u,v)}{\Delta b}\right| \tag{9}$$

其中，$a_b(u,v)$ 是子带 $b(LL,LH,HL,HH)$ 内的变换系数；$q_b(u,v)$ 是其量化值，Δb 是量化阶，可由

$$\Delta b=2^{R_b-\varepsilon_b}\left[1+\frac{\mu_b}{2^{11}}\right] \tag{10}$$

求出。R_b 取决于信源的编码位数，ε_b 和 μ_b 是分配给子带系数的指数和尾数的比特数。

第一、二级编码：经过量化后的系数要组织成为码块、位平面和编码通道．第一级实质就是对各种编码要素进行了熵编码，主要是基于上下文的 2 元算数编码．经过小波变换和量化，片分量矩阵成为了整数系数的子带矩阵．每个子带又要被划分为大小相同的矩形码块，码块的宽和高都需是 2 的整数次幂，典型值是 32×32 或 64×64，本实验采用 64×64 码块，每个码块将被独立编码．每个码块又可以分成位平面，即比特层．从最高有效位平面开始逐平面编码直到最低位平面。

第二级编码过程实际上是分层、打包、形成码流的过程。这层编码完成了 JPEG2000 标准具有的许多优良性能，如压缩质量和分辨率可分级性．分层装配的目的就是按照率失真最优的原则，选取合适的截断点截断每一分块的压缩码流，装配成具有分辨率可伸缩性或质量可伸缩性的满足预定编码长度的最终码流．编码时，对每一个截断点都根据率失真优化算法进行计算．然后将截断点和失真值以压缩的形式同码块位流保存在一起，形成码块的嵌入式压缩位流。

3 基于深度学习的图像压缩编码——自动编码器

神经网络训练的进步已经帮助改善了许多领域的性能，但是神经网络在有损图像压缩方面还没有超过现有的编解码器。自动编码器有潜力解决日益增长的对灵活的有损压缩算法的需求。根据不同的情况，需要不同计算复杂度的编码器和解码器。当从服务器向移动设备发送数据时，可能需要将功能强大的编码器与不太复杂的解码器配对，但是当向另一个方向发送数据时，要求就相反了。随着新技术的出现，计算能力和可用带宽的数量也会随着时间而变化。从存档的目的来看，编码和解码时间比流应用程序要少一些。最后，现有的压缩算法对于新媒体格式(如光场图像、360 视频或 VR 内容)可能远非最佳。虽然新编解码器的开发可能需要数年时间，但基于神经网络的更通用的压缩框架可能能够更快地适应这些不断变化的任务和环境。

不幸的是，有损压缩本质上是一个不可微的问题。具体地说，量化是压缩管道的一个组成部分，但它是不可微的，这使得训练神经网络来完成这项任务变得困难。自动编码器的目标往往是直接优化自动编码器产生的率失真权衡。研究者们提出了一种简单而有效的方法来处理基于四舍五入的量化的不可微性，并对生成的系数编码的不可微代价进行近似。

基于深度学习的图像压缩编码的网络架构通常包括以下几个部分：

1. 自编码网络；
2. 量化；
3. 熵编码、先验建模和码字估计；
4. 率-失真优化。

在介绍经典的自动编码器压缩框架之前，需要先明确几个概念：

1. 码字：编码后的比特流；
2. 码字大小：压缩比特存储空间；
3. bits per pixel (bpp): 图片存储中每个像素消耗的比特数;

4. PSNR: 是“Peak Signal to Noise Ratio”的缩写，即峰值信噪比，是一种评价图像的客观标准;
5. MS-SSIM：一种主观视觉效果评价指标。

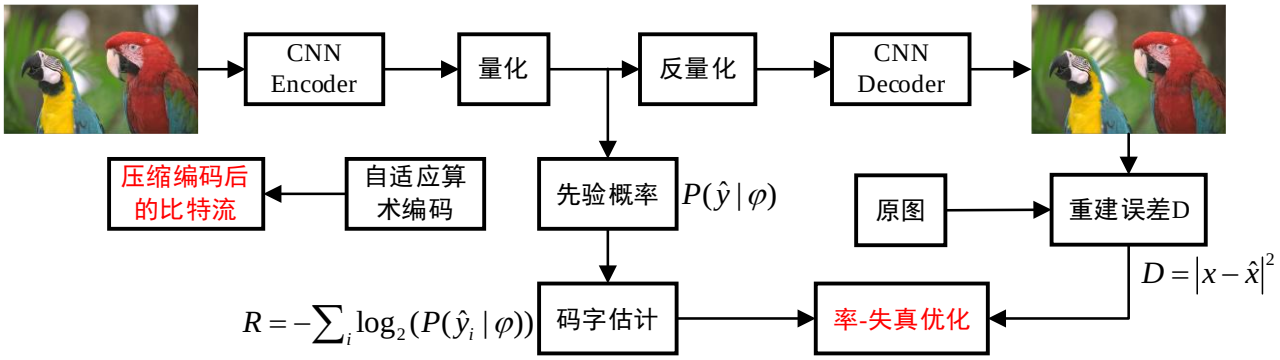


图 4 典型的深度学习图像压缩框架

典型的深度学习的图像压缩框架如图 4 所示，其中 CNN Encoder 被用作特征提取，实现对原始图像的特征变换与降维。输出的特征需要经过量化，量化是整个压缩编码过程中信息损失的主要来源。通常采用的量化方法有：

1）取整加随机噪声, (Lucas Theis, 2017)

$$\hat{y}\approx\lfloor y\rfloor+\varepsilon,\varepsilon\in\{0,1\} \tag{11}$$

2）训练时加随机噪声，测试直接取整 (Johannes Balle, 2016)

$$\hat{y}\approx y+u,u\in[0,1] \tag{12}$$

3）矢量量化, (Eirikur Agustsson, 2017)

$$\hat{y}\approx\sum_i c_i\varphi_i(\bar{y}) \tag{13}$$

4）二值量化, (George Toderici, 2017)

$$\hat{y}\approx\begin{cases}1, y>0\\0, y\le 0\end{cases} \tag{14}$$

量化后的熵编码是无损的，熵编码只是利用了量化后符号的统计信息实现了进一步的压缩，并不引入新的误差。所以，在实际验证该框架编解码性能的时候，不需要经过熵编码。但是，为了计算最终的码字速率，需要对码字进行估计，所以在训练该框架时，需要对量化后的符号进行先验建模、码字估计，得到码率，进而和失真函数 MSE 一起构成率-失真优化损失函数，来指导整个端到端自动编码器的学习。

4 JPEG 和 JPEG2000 的编程实现

4.1 JPEG 的编程实现

为了实现起来的方便，本实验仅以灰度图像为例，进行 JPEG 的压缩。同时，整个过程不存储压缩后的具体编码，仅从理论上计算编码长度、PSNR 和压缩率，即 JPEG 的压缩性能。具体实现流程如下：

实验前的准备

1. 灰度图片一张，选自 kodim 数据集，H=512,W=768；
2. codelength.mat, 即霍夫曼编码码长矩阵；
3. lighttable.mat, JPEG 标准亮度量化表；
4. zigzag.mat，快速 Z 型排序行向量。

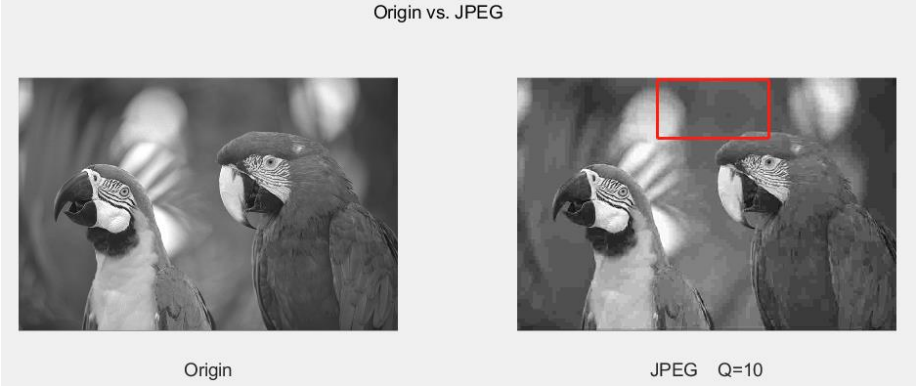
实验流程：

1. 图像分割：以 8x8 为最小单元分割，可分割成 $H\times W/8/8=6144$ 个方块，从上往下存储，得到 6144×8 的矩阵；
2. DCT 变换：对这 6144 个 8x8 的块分别进行 DCT，直接使用 MATLAB 的函数；
3. 量化：根据量化表对这 6144 个 8x8 的块进行量化，注意这一步在使用量化表之前，可以加入压缩质量控制变量 Q，根据 Q 对量化表进行一定的修改，可以控制 JPEG 的压缩质量；
4. 系数编码和熵编码：对量化后的值进行系数编码和熵编码，进一步压缩数据。注意这里并没有真正实现熵编码，只是根据理论计算了编码后的码长。

在上述编码过程中，量化一步做完后即可通过反量化、逆 DCT 获得原始图像，然后即可计算相应的压缩性能。实验结果如图 5 所示：


```
PSNR_JPEG:              32.682
Original Bit:          3145728 bit
Compressed Bit:       130782 bit
Compression Ratios:  24.0532
Pixel Depth:           0.3326 bpp
                          --Calculated by Tong
```

（a）




```
PSNR_JPEG:              29.9234
Original Bit:          3145728 bit
Compressed Bit:       88346 bit
Compression Ratios:  35.6069
Pixel Depth:           0.22468 bpp
                          --Calculated by Tong
```

（b）

图 5 不同质量因子下 JPEG 压缩性能：（a）Q=20 （b）Q=10，可以看到，Q=10 的图片质量明显不如 Q=20 的图片质量

如果直接采用 MATLAB 的 JPEG 编码工具函数 imwrite，也能够获得相应压缩质量的 JPEG 图片，本文对上述过程和 imwrite 的压缩进行了对比，结果如图 6 所示：


```
Original Bit:                3145728 bit
Compression Bit(self):       130782 bits
Compression Ratios(self):    24.0532
Pixel Depth(self):           0.3326 bpp
PSNR_JPEG(self):             32.682
Compression Bit(imwrite):    108160 bits
Compression Ratios(imwrite): 29.084
Pixel Depth(imwrite):        0.27507 bpp
PSNR_JPEG(imwrite):          34.4737
```

图 6 自己的程序和 MATLAB 函数 imwrite 的 JPEG 压缩性能对比，Q=20，分别对比了压缩率和 PSNR 等性能指标

可以看到，在相同的质量因子控制下，相比 imwrite，自己程序的压缩性能要差一些，说明自己的代码还有一些可以优化的空间。

4.2 JPEG2000 的编程实现

本实验基于 MATLAB 平台实现 JPEG2000 的压缩算法，实现的是 JPEG2000 的有损压缩方式。

在预处理部分，我们先进行了电平位移，然后进行颜色变换；

在小波变换部分，我们严格按照上文所描述的 Db9/7 提升滤波器进行正变换和反变换并获得了非常好的效果；

在量化部分，对于量化步长的确定，首先将参数分别设为 7 和 8．对于 Rb 的确定，根据码块中的（绝对）最大值与码块的子带信息分为了 12 种情况。

在 EBCOT 部分，我们首先将量化好后的数据进行分块，分成大小为 64×64 的子块，并通过元胞数组存储每个子块的信息（如级数，子带数，像素集等等），将元胞数组作为 EBCOT 系统的输入，每四行分为一个迭代进行三通道编码，将三个通道编码数据存放在一起输入的 $x.file$ 数组中，形成最后的码流。

之后是 Huffman 编码部分，将码流作为输入，由于 Huffman 编码的对象为文本类型或字符串类型，因此须将码流转换为字符串后再输入，这样以后就得到了最终的压缩码流。

恢复过程即为相应部分的逆过程的拼接，需要注意的是在电平位移的恢复环节与小波逆变换两个环节中有一些细节不能忽视．在小波逆变换的过程中，需要先将分块拼接起来再进行逆变换，否则会导致恢复出的图像块效应明显．其次，在电平位移的恢复过程中，要注意数值的溢出问题，应先将进行 double 型的运算，后将其转换为 uint8 型。

程序运行结果如图 7 所示。

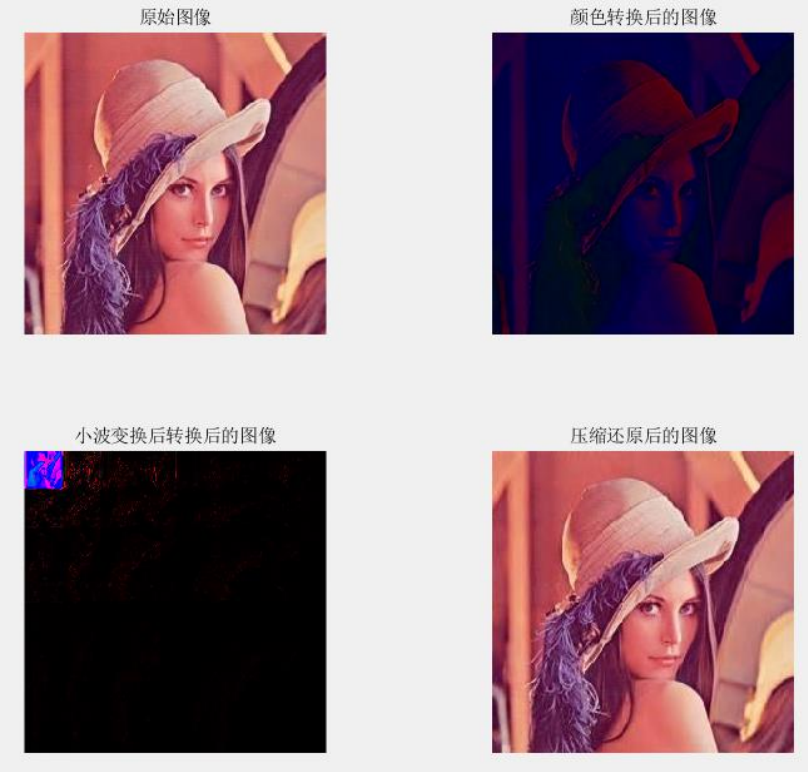


图 7 JPEG2000 恢复图像与原图像的比较结果比较

JPEG2000 比 JPEG 编码要复杂得多，所以对于 JPEG2000 的实现，只进行上述简单的操作。之后有需要对图像进行 JPEG 和 JPEG2000 压缩编码的地方，尽可能都是用 imwrite 实现。

4.3 基于自动编码器图像压缩的编程实现

以一个具体的自动编码器框架为例，如图 8 所示。跟第三部分所述的典型深度学习图像压缩框架大致相同，唯一不同的是，该框架增加了对图片重要性的考虑，因为我们的目标是一般的图像压缩，因此这里我们暂不考虑重要性的模块。

首先将待压缩图片输入 CNN Encoder，得到编码后的降维特征 $E(x)$，其中，CNN Encoder 包括三个卷积层和三个残差块以及相应的激活函数。卷积层进行下采样操作，Decoder 是 Encoder 得逆过程，为了实现上采样，本实验采用了一种深度到广度得变换方法。具体参考程序。

然后是四舍五入二值量化，得到 $B(E(x))$，其中

$$B(e_{ijk})=\begin{cases}1, & \text{if } e_{ijk}>0.5\\ 0, & \text{if } e_{ijk}\le 0.5\end{cases} \tag{15}$$

为了防止量化的不可微导致梯度无法回传，在梯度反向传播时，将量化函数用一个代理函数 $\tilde{B}(e_{ijk})$ 近似，其中

$$\tilde{B}(e_{ijk})=\begin{cases}1, & \text{if } e_{ijk}>1\\ e_{ijk}, & \text{if } 0<e_{ijk}<1\\ 0, & \text{if } e_{ijk}\le 0\end{cases} \tag{16}$$

将此二进制码元输入到 CNN Decoder，解码可得恢复后的图片。另外，整个网络的损失函数仍然是率失真函数，即优化目标为

$$L=\sum_{x\in\mathcal{X}}\{L_D(c,x)+\gamma L_R(x)\} \tag{17}$$

其中，c 是输入图像 x 经过编码、量化后的输出，$L_D(c,x)$ 代表失真损失，$L_R(x)$ 代表码率损失。他们分别定义如下：

$$L_D(c,x)=\left\|D(c)-x\right\|_2^2 \tag{18}$$

$$L_R(x)=\begin{cases}\sum_{i,j}\left(P(x)\right)_{ij}-r, & \text{if}\sum_{i,j}\left(P(x)\right)_{ij}>r\\ 0, & \text{otherwise}\end{cases} \tag{19}$$

阈值 r 可以根据给定压缩率的码长来设置。通过这种方式，我们的速率损失将惩罚码长大于 r，并使学习压缩系统达到在给定系统附近的可比压缩率。

利用松弛速率损失和梯度的直通式估计，整个压缩系统可以用 ADAM solver 进行端到端训练。在没有重要度映射的情况下，我们用预先训练好的参数在训练集 X 上初始化模型。对模型进行进一步的训练，学习速率为 1e−4,1e−5 和 1e−6。在每个学习率下，对模型进行训练，直到目标函数不下降为止。

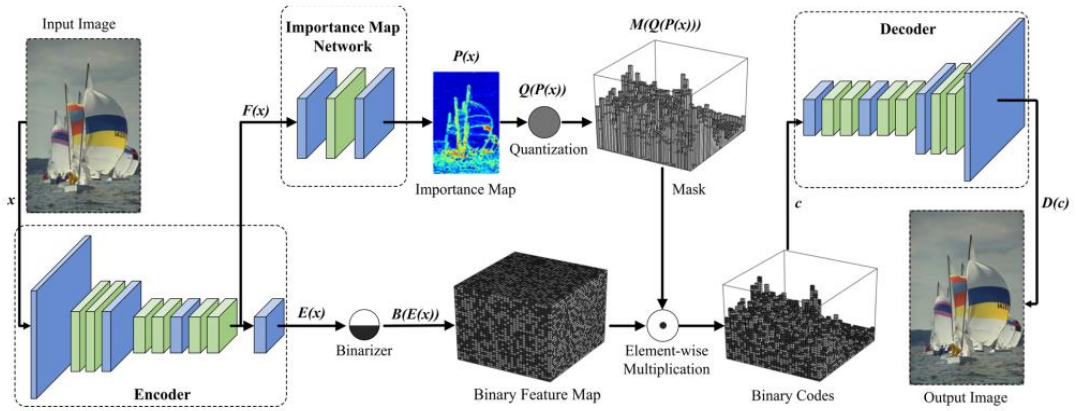


图 8 基于 CNN 的包含内容权重的图像压缩

训练到一定程度以后，测试该自动编码器压缩图像的性能，结果如下：

图 9 自动编码器恢复图像效果

直观看来，自动编码器能够将图像恢复得很好，但是有时候也会出现拼接造成的方框痕迹。

5 JPEG、JPEG2000 和自动编码器的压缩性能对比

对比压缩性能，往往要在相同的压缩率下，根据一些客观的评价指标来对比。常用的压缩率指标为 bpp，常用的图像感知质量评价指标为峰值信噪比（PSNR）和结构相似性（SSIM）。由于自动编码器射击训练过程，这里也增加了 MSE 这一评价指标。另外，为了跟实际的任务联系起来，考察恢复的图片在完成相应任务方面的性能。考虑了简单的分类任务，训练数据集采用 STL10。总体性能对比如表 1 所示。

表 1 JPEG、JPEG2000 和自动编码器性能对比

	1000 张图片测试准确率	100 张图片测试 SSIM	100 张图片测试 PSNR	压缩程度 bpp	loss(MSE)
PNG（原始图片格式）	95.30%	*	*	24	0
JPEG—Q=100	95.00%	0.9949	42.6575	6.662	0.00011618
JPEG—Q=75	**92.80%**	**0.9712**	**34.9074**	**1.736**	**0.00054921**
JPEG—Q=50	90.40%	0.9553	32.8474	1.298	0.00085318
JPEG—Q=25	82.20%	0.9276	30.5295	0.972	0.00136487
JPEG—Q=10	50.60%	0.8503	26.9303	0.689	0.00279378
JPEG2000—CR=1	95.20%	0.9987	51.1122	6.109	0.00000801
JPEG2000—CR=5	94.80%	0.9973	47.8884	4.501	0.00004184
JPEG2000—CR=10	94.80%	0.9877	40.7112	2.250	0.00018179
JPEG2000—CR=20	**92.30%**	**0.9641**	**35.0128**	**1.189**	**0.00066718**
JPEG2000—CR=25	89.00%	0.9512	33.4484	0.953	0.00095564
JPEG2000—CR=30	86.40%	0.9384	32.2386	0.792	0.00124839
自动编码器	**92.60%**	**0.9631**	**31.0572**	**2.000**	**0.00034132**

其中，Q 代表质量因子，CR 代表压缩率，第 1 列是不同的压缩编码方案，第 2 列是将恢复后的图片经过一个预先训练好的分类网络的分类结果，第 3 列和第 4 列是图片感知质量指标 SSIM 和 PSNR，第 5 列代表压缩程度 bpp，最后一列代表恢复后图片与原图的 MSE。

需要注意的是，此处的自动编码器求得的 bpp 尚未考虑熵编码，加上熵编码后，图像将得到进一步压缩，从而 bpp 也会更小。

6 总结

本文重点关注了静态图像压缩方法，对比了传统的压缩编码方法和近些年来基于深度学习的压缩编码方法的性能。

首先介绍了传统的 JPEG 压缩编码，JPEG 提出较早，实现起来并不复杂，而且有很好的性能，直到今天，在互联网上都应用广泛。在介绍了 JPEG 压缩的原理之后，又介绍了在其基础上发展出来的 JPEG2000，同样介绍了其原理。之后，鉴于近年来深度学习方法的流行，有不少研究者尝试采用深度学习的方法来实现端到端的图像压缩，进而实现端到端的通信。本文以一种典型的图像压缩框架为例，介绍了基于深度学习的图像压缩——自动编码器。在介绍完原理知后，分别用程序实现了相应的功能，从实现流程上去认识这些压缩方法，感受传统的图像变换和深度学习特征变换的区别与联系。最后，对比了各种方案的压缩性能，结果显示，基本的自动编码器已经具备和 JPEG、JPEG2000 相媲美的性能。

参考文献

[1] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. IEEE Signal processing magazine, 18(5):36–58, 2001.

[2] G. K. Wallace. The JPEG Still Picture Compression Standard. IEEE Transactions on Consumer Electronics, 1991.

[3] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhao, “Learning convolutional networks for content weighted image compression,” in Proc. IEEE Conf. Comput. Vision Pattern Recognit., 2018, pp. 3214–3223.

[4] L. Theis, W. Shi, A. Cunningham, and F. Huszar. Lossy image compression with compressive autoencoders. arXiv preprint arXiv:1703.00395, 2017.

[5] J. Balle, V. Laparra, and E. P. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In Picture Coding Symposium, 2016.

[6] 周雷. 深度学习在压缩算法上的应用. 图鸭信息科技. 2018