参赛队员姓名：赵海萌

中学：上海市上海中学

省份：上海

国家/地区：中国

指导教师姓名：沈孝山

论文题目：CAE-ADMM: Implicit Bitrate Optimization via ADMM-based Pruning in Compressive Autoencoders

CAE-ADMM: 图像压缩自编码器中基于 ADMM 剪枝的隐式比特率优化方法

CAE-ADMM: Implicit Bitrate Optimization via ADMM-based Pruning in Compressive Autoencoders

赵海萌

shxx3528zhao@163.com

上海市上海中学

摘要

近年来的研究中，由深度神经网络构成的自编码器模型在图像压缩领域取得了巨大的成功，已经成为图像压缩前沿的几大最有潜力的研究方向之一。基于 CNN 和 RNN 的图像压缩自编码器均取得了同等甚至优于现有的通用图像编码器 (JPEG、JPEG 2000、BPG、WebP 等) 的性能。为了处理图像压缩率与失真率之间的关系，现有的研究采取多种方法对图像编码量化后的信息熵进行近似和估计，并依此在优化的目标函数中表征图像的压缩率（比特率）。而本文避开了额外训练信息熵估计器的麻烦，引入 ADMM 算法 (Alternating Direction Method of Multipliers) 直接对表示层以及卷积层的结构进行剪枝，隐式地引导自编码器在优化图像重构质量的同时优化比特率，并基于此提出了 CAE-ADMM 模型 (ADMM-pruned Compressive AutoEncoder)。实验表明，CAE-ADMM 在图像重构质量和压缩率上均超越了现有的图像压缩算法。

关键词—有损图像压缩，深度学习，自编码器，非凸优化，神经网络剪枝

Abstract

We introduce ADMM-pruned Compressive AutoEncoder (CAE-ADMM) that uses Alternative Direction Method of Multipliers (ADMM) to optimize the trade-off between distortion and efficiency of lossy image compression. Specifically, ADMM in our method is to promote sparsity to implicitly optimize the bitrate, different from entropy estimators used in the previous research. The experiments on public datasets show that our method outperforms the original CAE and some traditional codecs in terms of SSIM/MS-SSIM metrics, at reasonable inference speed.

Index Terms—autoencoder, lossy image compression, neural network pruning, bitrate optimization

目录

1 引言 3

2 相关研究 4

3 CAE-ADMM 模型 5

3.1 编码器 E 和解码器 D 5

3.2 量化器 Q 6

3.3 引入 ADMM 剪枝方法解决比特率优化问题 6

3.4 组合起来 9

3.5 CAE-ADMM 与现有模型的差异 10

4 实验 10

4.1 模型架构 10

4.2 目标函数 d 的选择 12

4.3 模型训练 12

4.4 数据集及其预处理 12

4.5 实验结果及讨论 12

5 结论 15

6 附录：已预印版本 arXiv:1901.07196 20

7 致谢 26

1 引言

自 Theis 等人提出的图像压缩自编码器 CAE (Compressive AutoEncoder)[1] 之后，自编码器作为一种简单高效的神经网络模型在有损图像压缩领域取得了极大的成功。基于 CNN 的图像压缩自编码器一经提出便达到了优于 JPEG、JPEG 2000 等传统图像压缩算法 [2] 以及基于 RNN 的神经网络方法的效果 [3][4]。Mentzer 等人在此基础上引入了 3D-CNN 的上下文模型，使图像压缩自编码器的性能超越了工业级的图像压缩算法 BPG[5]。此类基于深度神经网络的图像压缩算法不同于传统图像压缩算法，能够自主学习所给图像的特征以及其所适合的压缩方式。正是这样的高自适应性使其能达到更高的压缩率。

图像压缩自编码器 CAE，亦即编码器 E 及解码器 D，的训练过程可以转化为一个优化问题，即对图像失真率以及图像编码比特数的最小化。有损图像压缩面临着失真率以及压缩率的权衡问题，因此可以将上述优化问题表述为：

$$\min_{E,D}\ d + \beta R,$$

其中 d 表示重构图像与原图像之间的差距，R 表示图像编码比特数，而 $\beta > 0$ 则控制上述两个因素之间的平衡。

解决这个优化问题的过程中会遇到许多困难，其中最为重要的一个是如何表征图像编码比特数 R。现有的研究对此均采用熵估计的方法：依据信息论，已量化编码的信息熵 H 可以作为 R 的一个合理表征，于是我们可以额外训练一个熵估计器，用于估计已量化编码的信息熵 H, 并用 H 替代 R，将优化任务转变为 $\min\ d + \beta H$。在这一问题中，现有的研究分别利用了 GSM (Gaussian Scale Mixtures)[1]、3D-CNN 上下文模型 [5] 等方法对 H 进行建模。然而这样的方法对 R 的优化不直接、不准确，而且需要额外预训练熵估计器，较为繁琐粗糙。

因此本文针对 R 的优化，提出利用 ADMM (Alternating Direction Method of Multipliers) 算法对 CAE 的表示层进行剪枝（如图1所示），即直接减小 R，避开了额外训练信息熵估计器的麻烦，并遵循训练、剪枝、重训练的顺序，迭代地对 CAE 进行训练 (优化 d) 和剪枝 (优化 R)，直至达到目标要求。而在此之前，ADMM 算法已经在对一般深度神经网络进行剪枝的问题中取得了极好的性能 [6][7]。基于此，本文提出了 CAE-ADMM (Compressive AutoEncoder with Pruning) 模型。相较于现有的 CAE 模型，本文的模型显得更为简单直接，更易实现且参数量更小。实验中，我们的模型在 MS-SSIM (Multi-scale Structural Similarity Index)[8]、SSIM[9] 等指标下均超越了现有的图像压缩算法。

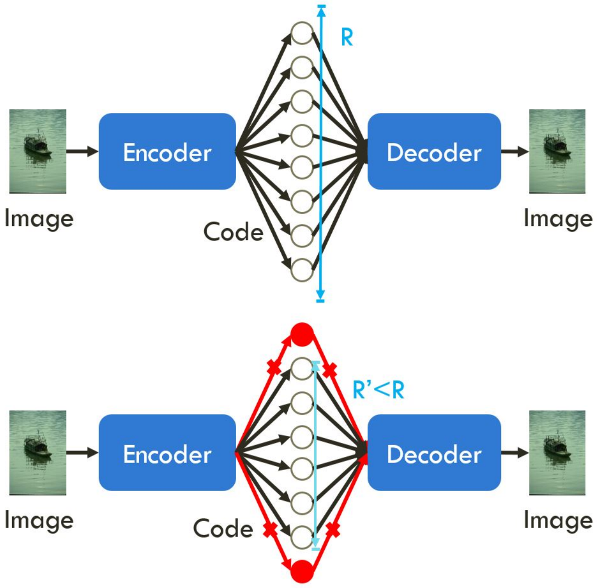


图 1: 剪枝过程示意图

2 相关研究

利用深度神经网络进行有损图像压缩的方法，在近年的研究中，因其优秀的表现而引起了广泛的关注。目前用于有损图像压缩的神经网络结构主要有两种：压缩自编码器 CAE 结构 [10][1][11][12][13] 以及基于循环神经网络 RNN 的结构 [3][4]。此类神经网络方法普遍以优化 $d+\beta R$ 为目标，采用 MSE (Mean-Squared Error)、SSIM、MS-SSIM 等指标表征 d，额外训练熵估计器估计 H 并借以表征 R。常见的熵估计器有生成式模型 [11]、GSM[1]、上下文模型 [5] 等等。其他的用于有损图像压缩的神经网络方法还包括：生成式对抗模型 [14]、GDN (Generalized Divisive Normalization)[11] 等等。

本文的研究避开了现有的熵估计的繁琐方法，转而采用更为直接有效的剪枝方法。这种剪枝方法受到了神经网络压缩领域研究的启发。由于计算资源的限制，神经网络在实际应用中的一大困难在于其庞大的参数量，以及由此所导致的训练的困难。因而自动改进网络结构、减少网络参数量、加快训练速度的方法一直以来受到了持续而广泛的关注和研究。其中一种简单而又有效的方法是由 Han 等人于 2015 年首次提出的剪枝方法 [15]。这种方法迭代地删除网络中不重要的权重并进行重训练，从而产生更为紧凑的网络结构。顺着这种思路，进一步的研究对剪枝方法不断进行了改进，并取得了很好的网络压缩效果 [16][17][18][19][20]。Zhang 等人于 2018 年对此类剪枝方法提出了基于 ADMM 的理论解释

以及系统的剪枝方法 [7][6]，并在 ImageNet 数据集及 MNIST 数据集上均达到了目前最好的网络压缩效果。其他的神经网络架构搜索的方法还包括强化学习 [21]、遗传算法 [22] 等等。

3 CAE-ADMM 模型

一个基本的图像压缩自编码器 CAE 由三部分组成：编码器 E、解码器 D 和量化器 Q，

$$E : \mathbb{R}^n \to \mathbb{R}^m,$$
$$D : \mathbb{R}^m \to \mathbb{R}^n,$$
$$Q : \mathbb{R} \to \mathbb{Z}.$$

编码器 E 把原始图像数据 $\mathbf{x}$ 映射为一组隐式的表示形式 $\mathbf{z} \coloneqq E(\mathbf{x})$. 量化器 Q 把表示 $\mathbf{z}_i$ 映射到 $\mathbb{Z}$, 从而得到原始图像压缩量化后的编码 $\hat{\mathbf{z}}_i \coloneqq Q(\mathbf{z}_i)$. 紧接着，解码器 D 从量化后的编码 $\hat{\mathbf{z}}$ 中试图重构出原始的图像 $\hat{\mathbf{x}} \coloneqq D(\hat{\mathbf{z}})$.

为了使重构出的图像 $\hat{\mathbf{x}}$ 与原始图像 $\mathbf{x}$ 尽可能接近，同时尽可能减小编码的比特数 R，我们需要解决以下优化问题：

$$\min_{E,D} \; d(\mathbf{x}, \hat{\mathbf{x}}) + \beta R(\hat{\mathbf{z}}). \tag{1}$$

接下来，我们会就编码器 E、解码器 D、量化器 Q 如何选择，(1) 式所述的优化问题如何解决，以及如何由此构建出 CAE-ADMM 模型等问题进行详细的讨论。

3.1 编码器 E 和解码器 D

与 Theis 等人提出的 CAE 相似 [1]，我们采用卷积神经网络 CNN[23] 作为编码器 E、解码器 D 的基础，从而构成一组卷积自编码器。为了避免网络的深度所带来的梯度问题，我们使用残差块 [24] 作为该自编码器的基本单位，从而保证反向传播的过程中不会产生梯度消失等问题。编码器 E 和解码器 D 的结构基本镜像对称，区别在于编码器 E 中，卷积层自动地承担了下采样的工作；而在解码器 D 中，我们则需要对多通道小尺寸张量的参数进行重排，使其成为少通道大尺寸的张量，以完成上采样的工作，这一步骤可以应用 Shi 等人 [25] 提出的 sub-pixel convolutional layer。至此，我们构建出了以残差块为主体

的编码器 E 和解码器 D，通过适当的训练，它们可以有效的学习出图像的潜在表示形式 $\mathbf{z}$，并学会如何从量化后的 $\hat{\mathbf{z}}$ 中重构出图像 $\hat{\mathbf{x}}$。

值得注意的是，由于编码器 E 由卷积神经网络构成，因此编码器 E 所学习到的 $\hat{\mathbf{z}}$ 事实上是一组特征图 (feature map)$\{\hat{\mathbf{z}}_i\}$。Matthew 等人 [26] 对卷积层可视化的研究表明，卷积层产生的特征图事实上反映了网络对于图像特征的认识。因此这为我们将在章节3.3中讨论的通过对 $\hat{\mathbf{z}}$ 剪枝来优化 R 的方法提供了更深层的解释：这样的剪枝步骤迫使编码器 E 抛弃重要性较低的特征，保留重要性较高的特征，从而达到在对重构失真率影响较小的情况下尽可能减小 R 的目的。

3.2 量化器 Q

量化器 Q 的构造方式有很多，为了尽可能地减小计算资源消耗并保证实施的简洁，我们采用 Theis 等人 [1] 提出的随机量化方法, 该方法受 Toderici 等人 [3] 所采用的随机二值化方法启发，将量化器 Q 定义为：

$$Q(\mathbf{z}_i) \coloneqq \lfloor \mathbf{z}_i \rfloor + \epsilon,$$

其中 $\epsilon \in \{0, 1\}$ 控制是向上还是向下取整，且满足 $\epsilon = 1$ 的概率 $P(\epsilon = 1) = \mathbf{z}_i - \lfloor \mathbf{z}_i \rfloor$.

由于上述 Q 并非可导函数，因此在反向传播时，我们用其数学期望的梯度代替其梯度：

$$\frac{\partial}{\partial \mathbf{z}_i} Q(\mathbf{z}_i) \coloneqq \frac{\partial}{\partial \mathbf{z}_i} \mathbb{E}(Q(\mathbf{z}_i)) = \frac{\partial}{\partial \mathbf{z}_i} \mathbf{z}_i = 1$$

这样设计的量化器 Q 使得我们在反向传播时可以将量化操作直接视为恒等映射。在 PyTorch 实现过程中，我们便可以调用 torch.no_grad() 来节省计算资源。

3.3 引入 ADMM 剪枝方法解决比特率优化问题

在 (1) 式所述的优化问题中，我们需要选择合适的 $d(\mathbf{x}, \hat{\mathbf{x}})$ 和 $R(\hat{\mathbf{z}})$ 的表示形式。$d(\mathbf{x}, \hat{\mathbf{x}})$ 通常选用 MSE[1]、MS-SSIM[5] 等指标衡量图像失真率。而在此前的研究中，$R(\hat{\mathbf{z}})$ 往往采用额外训练熵估计器的方法，利用 $\hat{\mathbf{z}}$ 的信息熵 $H(\hat{\mathbf{z}})$ 来代替，即 $R(\hat{\mathbf{z}}) \coloneqq H(\hat{\mathbf{z}})$.

上述的熵估计法需要额外预训练熵估计器，较为繁琐，而且对 R 的优化也不直接。针对这个问题，我们可以尝试重新表述 (1) 式所述的优化问题。首先，我们可以改写 $R(\hat{\mathbf{z}})$:

$$R(\hat{\mathbf{z}}) = \mathrm{card}(\hat{\mathbf{z}}) = \mathrm{card}(\mathbf{z}) = \mathrm{card}(E(\mathbf{x})),$$

其中 card$(\cdot)$ 表示其非零数的个数。如果我们想要达到的压缩效果是 $\mathbf{z}$ 的非零数个数小于等于某个常数 ℓ, 那么所有满足要求的编码器 E 构成了一个非凸集合 $\mathbf{S} = \{E \mid \mathrm{card}(E(\mathbf{x})) \leq \ell\}$. 于是 (1) 式所述的优化问题可以进一步表述为：

$$\begin{aligned} &\min_{E,D}\ d(\mathbf{x}, \hat{\mathbf{x}}), \\ &\text{s.t. } E \in \mathbf{S}. \end{aligned}$$

我们可以通过定义 $\mathbf{S}$ 的示性函数 $g(\cdot)$ 来改写上述非凸优化问题：

$$\min_{E,D}\ d(\mathbf{x}, \hat{\mathbf{x}}) + g(E(\mathbf{x})),$$

其中 $g(\cdot)$ 定义为

$$g(E(\mathbf{x})) \coloneqq \begin{cases} 0 & \text{if } \mathrm{card}(E(\mathbf{x})) \leq \ell, \\ +\infty & \text{otherwise.} \end{cases}$$

为了解决这个问题，我们可以应用 ADMM 算法 [27]。首先需要将上述问题改写为 ADMM 可以解决的形式：

$$\begin{aligned} &\min_{E,D}\ d(\mathbf{x}, \hat{\mathbf{x}}) + g(\mathbf{Z}), \\ &\text{s.t. } E(\mathbf{x}) - \mathbf{Z} = 0. \end{aligned}$$

ADMM 算法引入对偶变量 $\mathbf{U}$ 以及惩罚因子 $\rho > 0$，并将这个优化问题分解为两个子问题，并迭代地解决这两个子优化问题，直到收敛 [27]。第一个子问题是：

$$\min_{E,D}\ d(\mathbf{x}, \hat{\mathbf{x}}) + \frac{\rho}{2}\|E(\mathbf{x}) - \mathbf{Z}^k + \mathbf{U}^k\|_F^2,$$

其中 k 为迭代次数，$\|\cdot\|_F^2$ 为 Frobenius 范数。该子问题相当于仅在对 $d(\mathbf{x}, \hat{\mathbf{x}})$ 优化的基础上增加了一项 L_2 正则化项, 因而可以很容易地被梯度下降算法解决。第二个子问题是：

$$\min_{Z}\ g(\mathbf{Z}) + \frac{\rho}{2}\|E^{k+1}(\mathbf{x}) - \mathbf{Z} + \mathbf{U}^k\|_F^2.$$

Boyd 等人 [27] 导出了该子问题的解：

$$\mathbf{Z}^{k+1} = \Pi_{\mathbf{S}}(E^{k+1}(\mathbf{x}) + \mathbf{U}^k),$$

其中 $\Pi_{\mathbf{S}}$ 为 $\mathbf{S}$ 上的欧几里得投影。一般来说，计算非凸集上的欧几里得投影是很困难的。但是 Boyd 等人 [27] 证明了，在 $\mathbf{S} = \{E \mid \mathrm{card}(E(\mathbf{x})) \leq \ell\}$. 的特殊结构下，上述问题的解就是保留 $E^{k+1}(\mathbf{x}) + \mathbf{U}^k$ 中最大的 ℓ 项，并将其余项均置为零。最后, 我们按照以下规则更新对偶变量 $\mathbf{U}$：

$$\mathbf{U}^{k+1} = \mathbf{U}^k + E^{k+1}(\mathbf{x}) - \mathbf{Z}^{k+1}.$$

上述三步构成了 ADMM 算法的一次迭代。整个利用 ADMM 算法解决 (1) 式所述的优化问题的过程如 Algorithm 1 所示。

Algorithm 1 Pruning of $E(\mathbf{x})$ Based on ADMM

```
Input:
  $\mathbf{x}$: A batch of input images;
  $E, D$: The encoder and decoder;
  $\ell$: The expected number of non-zero elements in $E(\mathbf{x})$;
  $k_m$: Max number of iterations.
Output:
  $E, D$: Trained encoder and decoder;
  $\mathbf{U}, \mathbf{Z} \Leftarrow$ zeroes with the same shape as $E(\mathbf{x})$
  for $1 \leq k \leq k_m$ do
    $E, D \Leftarrow \arg\min_{E,D}\ d(\mathbf{x}, \hat{\mathbf{x}}) + \frac{\rho}{2}\|E(\mathbf{x}) - \mathbf{Z} + \mathbf{U}\|_F^2$;
    $\mathbf{Z} \Leftarrow$ keep the $\ell$ largest elements in $E(\mathbf{x}) + \mathbf{U}$ and set the rest to 0;
    $\mathbf{U} \Leftarrow \mathbf{U} + E(\mathbf{x}) - \mathbf{Z}$.
  end for
  return  $E, D$
```

值得注意的是，上述基于 ADMM 的剪枝算法并不能保证 $E(\mathbf{x})$ 最终仅剩小于等于 ℓ 个非零元素。在算法收敛时，$E(\mathbf{x})$ 中的许多元素会极为接近零，但并非严格为零。因此在 ADMM 迭代完成后，我们需要保留 $E(\mathbf{x})$ 中的 ℓ 个最大的元素，并将其余元素置为零，即强制剪除近零元素。为了减小这一步强制剪除对重构图像的影响，我们还需要在保持这些项为零的情况下对模型进行重训练，重训练只对 $\min_{E,D}\ d(\mathbf{x}, \hat{\mathbf{x}})$ 进行优化。

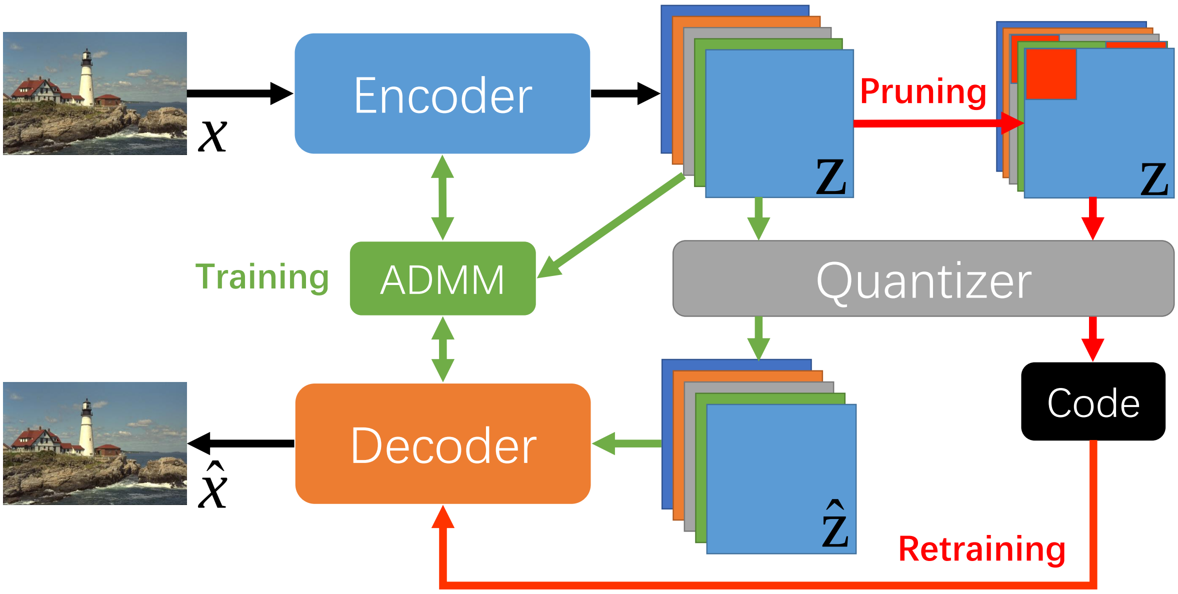


图 2: CAE-ADMM 模型的训练策略。其中 Encoder 和 Decoder 均采用基于 CNN 的残差块构建，Code 为最终的压缩后编码。ADMM 训练过程中，黑色和绿色通路激活，ADMM 算法在最小化重构失真率的同时对 $\mathbf{z}$ 剪枝，迫使其稀疏化。ADMM 训练结束后，黑色和红色通路激活，对 $\mathbf{z}$ 中的近零元素强制剪除，开始重训练。

3.4 组合起来

将前几节所述的内容组合起来，我们便得到了基于 ADMM 剪枝的图像压缩自编码器 CAE-ADMM(Compressive AutoEncoder with Pruning)，它的基本结构以及运作方式如图2所示。

原始图像 $\mathbf{x}$ 经过由卷积残差块构成的编码器 E 编码，转化为一组潜在表示形式 $\mathbf{z}$。在 ADMM 训练阶段，黑色和绿色通路激活，特征图 $\mathbf{z}$ 经量化器 Q 量化后得到 $\hat{\mathbf{z}}$，输入同样由卷积残差块构成的解码器 D，解码器 D 从中重构出图像 $\hat{\mathbf{x}}$，ADMM 算法迭代地最小化重构失真率 $d(\mathbf{x}, \hat{\mathbf{x}})$，并对 $\mathbf{z}$ 剪枝，迫使其稀疏化。ADMM 训练阶段结束后，绿色通路关闭，红色通路开启，进入重训练阶段。此时由于 ADMM 算法并不能保证 $\mathbf{z}$ 中仅剩余小于等于 ℓ 个非零元素，只能使其他元素接近于零，因此需要对 $\mathbf{z}$ 中的近零元素强制剪除，再

经量化器 Q 量化后，得到最终编码形式，并将此最终编码形式输入解码器 D 中，得到重构图像 $\hat{\mathbf{x}}$。为了减小强制剪除对重构图像的影响，此时的重训练便只需优化 $\min_{E,D}\ d(\mathbf{x},\hat{\mathbf{x}})$。遵循训练、剪枝、重训练的步骤反复训练，可以得到同时具有高图像重构质量以及高压缩率的图像编码。这便是本文所提出的 CAE-ADMM 模型的基本架构以及训练方法。

3.5 CAE-ADMM 与现有模型的差异

现有的模型普遍采用预训练熵估计器的方法，对 $\mathbf{z}$ 的信息熵 H 进行估计，并作出一系列的近似，从而用 H 直接代替 R。Theis 等人 [1] 训练了一组 GSM 来估计 H，而 Mentzer 等人 [5] 则受上下文模型的启发，采用优化 3D-CNN 的交叉熵的方法来间接优化 H。

而本文提出的 CAE-ADMM 则摒弃了这种方法，受到 ADMM 算法在神经网络参数剪枝中的成功应用的启发，转而利用 ADMM 直接对 $\mathbf{z}$ 剪枝，迫使其稀疏化，从而直接对 R 进行优化。值得注意的是，由于编码器 E 是卷积神经网络，因而 $\mathbf{z}$ 是原图像的一组特征图，于是这样的剪枝方法被赋予了更深层次的含义，即迫使编码器抛弃图像中重要性较低的特征，保留图像中重要性较高的特征，从而能够在增大压缩率的过程中更好地抓住图像特征。

4 实验

4.1 模型架构

与 Theis 等人的模型 [1] 相似，CAE-ADMM 由基于 CNN 和残差块的编码器、解码器组成，如图3所示。原始图像首先经过 3 次下采样，每次包含下采样卷积、批归一化 BN 层和 PReLU 激活函数；接下来经过 15 层 Bottleneck 残差块，其中每次卷积后均连接 BN 层和 PReLU；最后再经过两层下采样卷积，得到 $\mathbf{z}$。

接下来经过量化器得到 $\hat{\mathbf{z}}$，量化器的梯度如小节3.2所述，设置为 1。

量化后的编码进入解码器，解码器的架构与编码器基本镜像。编码首先通过两层上卷积 Sub-Pix 层，每层包含卷积、BN 层、PixelShuffle 和 PReLU 激活函数；然后经过 15 层 Bottleneck 残差块，内部结构与编码器中的相同；最后再经过两层上卷积 Sub-Pix 层，最后通过 Tanh 激活函数，使激活值限制在 $(-1,1)$ 之间，最后将其线性映射至 $(0,1)$ 之间。

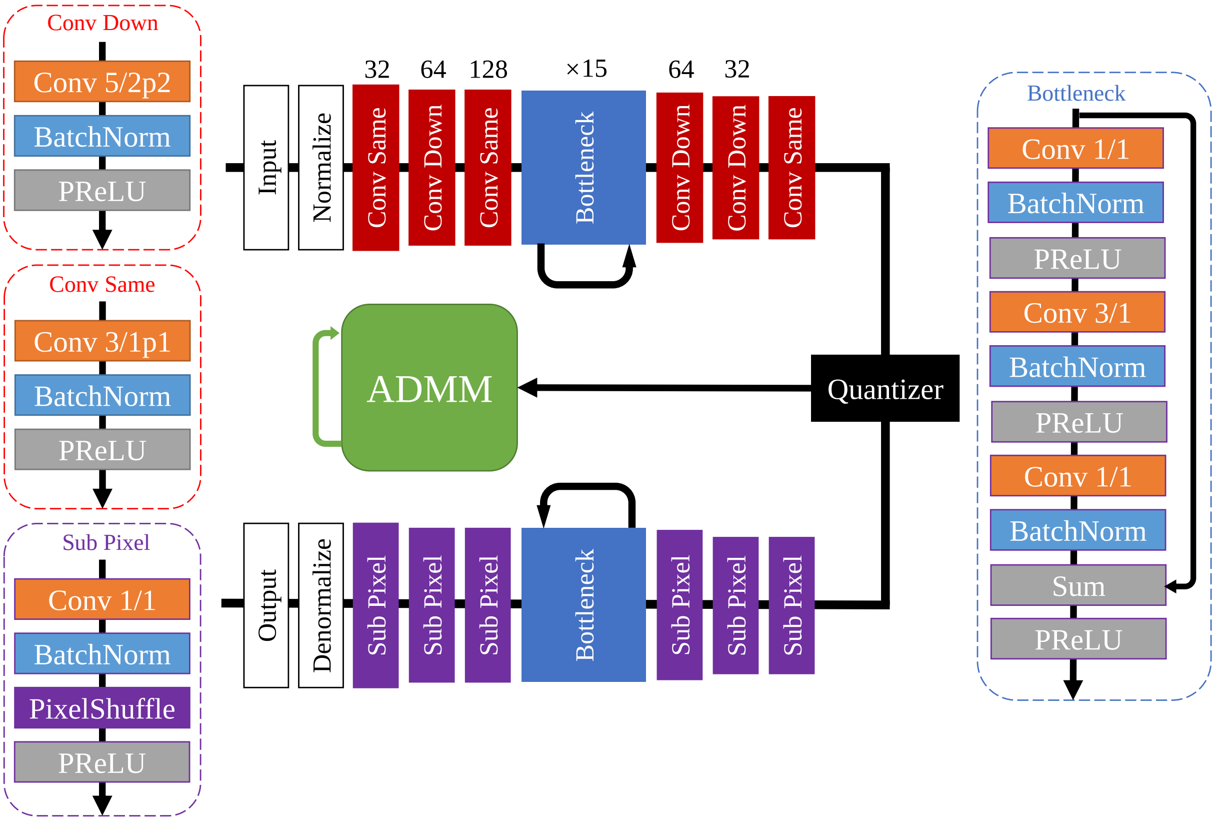


图 3: CAE-ADMM 的详细模型架构. “Conv k/spP” 表示卷积核大小为 $k \times k$、步长为 s，并采用大小为 P 的镜像 Padding 的卷积层, “Conv Down” 表示将宽和高减半的卷积层.

4.2 目标函数 d 的选择

目前常用的 d 的选择包括 MSE[1]、MS-SSIM[5] 等。我们尝试了多种设计后，最终选择 $d(\mathbf{x},\hat{\mathbf{x}}) = 100\cdot(1-\text{MS-SSIM}(\mathbf{x},\hat{\mathbf{x}}))+100\cdot(1-\text{SSIM}(\mathbf{x},\hat{\mathbf{x}}))+(45-\text{PSNR}(\mathbf{x},\hat{\mathbf{x}}))+\text{MSE}(\mathbf{x},\hat{\mathbf{x}})$。所有模型均以此为目标函数 d 训练。

4.3 模型训练

我们采用 Adam[28] 作为优化器，Batch Size 设置为 32，对小节3.3中 ADMM 优化问题的第一步进行优化。初始学习率设置为 $4\cdot10^{-3}$，并随着训练过程动态衰减：每出现 10 个 Epoch 平均损失函数不下降，学习率减半。第一个 Epoch 先进行 Warm-Up，仅使用 MSE 作为目标函数，以避免 MS-SSIM 指标在训练初期可能出现的异常值。Weight Decay 设置为一个较小值 $1\cdot10^{-10}$。每经过 20 个 Epoch，执行一次 ADMM 算法的第二、第三步，第二步中的保留元素比例设置为 10%。对于不同 bpp 的模型，我们修改编码器的最后一层及解码器的第一层的结构，进行 fine-tune。所有代码在 PyTorch 框架下实现，每个模型在 4 块 NVIDIA GeForce GTX 1080Ti GPU 上并行化训练了 300 个 Epoch，并在 GitHub 上开源[1]。

4.4 数据集及其预处理

我们选择 BSDS500[29] 作为训练数据集，包含 500 张 481×321 的图片。每次输入网络时图像首先被随机裁剪出一块 128×128 大小的 Crop，随机进行水平和竖直翻转，最后归一化至 $(0,1)$。对于测试集，我们采用图像压缩领域内通用的测试数据集 Kodak PhotoCD[2]，包含 24 张 768×512 的照片。

4.5 实验结果及讨论

图4展示了不同方法在 Kodak 测试集上的平均表现，图像重构质量分别用 SSIM、MS-SSIM 表征，压缩效果用单位像素比特数 bpp 表征。SSIM、MS-SSIM 的计算采用了开源实现[3]，其他方法的数据取自 Theis 等人的论文 [1]。由于较高 bpp 段不同方法表现相近，

[1] https://github.com/JasonZHM/CAE-ADMM

[2] http://r0k.us/graphics/kodak/

[3] https://github.com/jorge-pessoa/pytorch-msssim

不同于他们论文中选取的 bpp 范围 $[0, 3]$，我们选取了较低的 bpp 范围 $[0, 1]$，用以更好地展现不同方法之间的差异。图中可以看见，除了在极低 bpp 段我们的方法的 SSIM 指标下滑速率加大之外，我们的方法在 SSIM、MS-SSIM 两个指标下的表现均超越了现有的方法，尤其是我们所改进的对象 CAE。值得注意的是，图中蓝线所示的 Toderici 等人的方法同样没有采用熵估计器，而是采用了 RNN 架构。而我们的方法在使用剪枝方法替代熵估计器后，取得了超越前两者的表现，从而证明了我们所引入的剪枝方法的有效性。

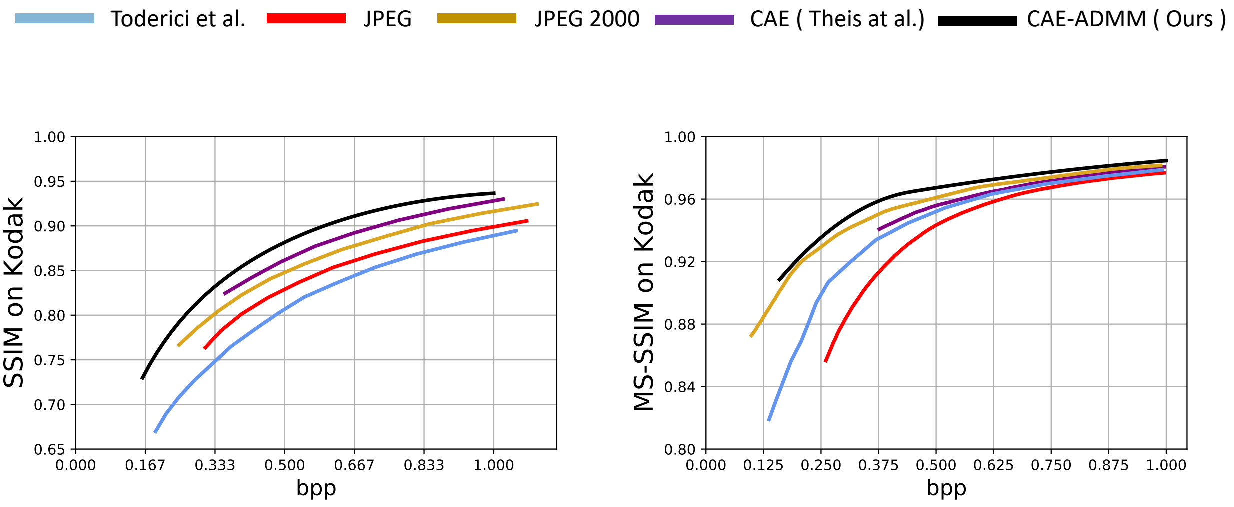


图 4: 不同方法在 Kodak 测试集上的表现，分别用 SSIM、MS-SSIM 表征（均为越接近 1 越好），压缩率用 bpp（单位像素所需比特数）表征。其中，Toderici et al. 使用 RNN 结构，并未使用熵估计器；CAE (Theis et al.) 采用熵编码器；CAE-ADMM (Ours) 使用剪枝方法替代熵估计器。

图6是使用不同方法的实际压缩效果图，左上是原图，左下是本文提出的方法 CAE-ADMM，右上是 JPEG，采用 libjpeg 实现[4]，右下是 JPEG 2000，采用 Kakadu 的实现[5]。在 bpp 0.3 的压缩率下，我们可以看出 JPEG 已经出现了严重的质量问题，JPEG 2000 及 CAE-ADMM 表现仍然稳定。仔细观察图像右方的云彩区域，可以看出 CAE-ADMM 比 JPEG 2000 质感更接近原图，也更清晰。在 bpp 0.5 的压缩率下，三者均表现稳定，没有出现严重的问题。CAE-ADMM 的水面颜色比 JPEG 和 JPEG 2000 更接近原图，其他细

[4] http://libjpeg.sourceforge.net/

[5] http://kakadusoftware.com/

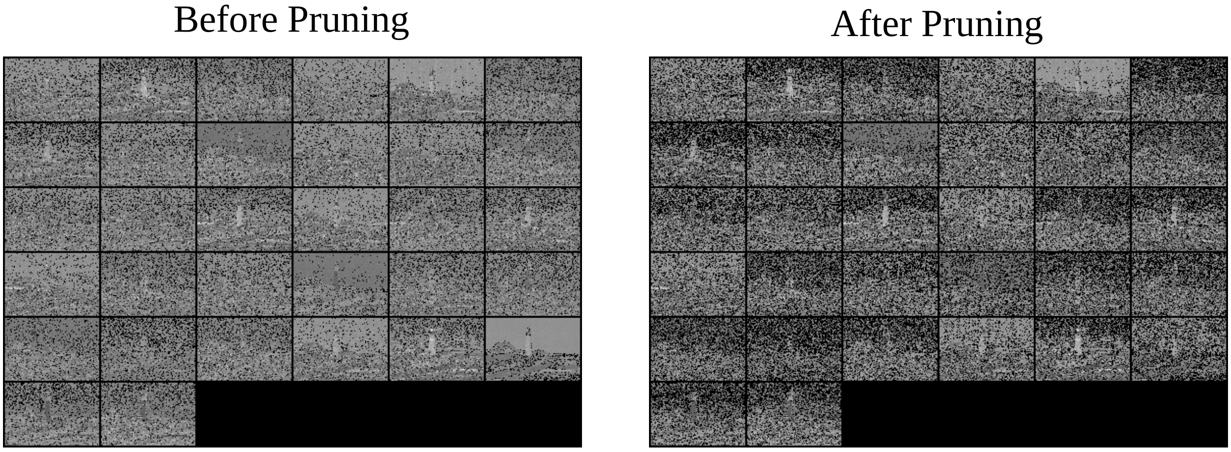


图 5: 剪枝前后模型所产生的紧凑编码的可视化，原图像为 *kodim21*，即图6(a)。为了清晰，其中的零元素已被标黑。

Model	bpp	ratio of zeros
Before pruning	1.684 ± 0.012	$7.80\% \pm 3.44\%$
After pruning	1.257 ± 0.011	$17.65\% \pm 4.90\%$

表 1: 剪枝前后的 bpp 以及 $\hat{\mathbf{z}}$ 中零元素的比例的对比。数据在混合测试集上进行统计，并建立了 95% 置信区间。

节没有很大差异。总体来看，CAE-ADMM 在较低比特率下稳定性更好，且压缩表现在不同比特率下均优于 JPEG、JPEG2000。

我们还进行了 Ablation Study，单独研究 ADMM 剪枝算法在模型中的有效性。我们用相同的训练策略分别训练了两个模型，一个采用了 ADMM 剪枝算法，一个没有。由于 [1] 的代码并未开源，因此无法与其进行比较。然后我们在一个混合测试集（由 Urban100 [30], Manga109 [30] 和 Kodak PhotoCD 中 768 × 512 的切片组成）上统计了平均的 bpp 以及 $\hat{\mathbf{z}}$ 中零元素的比例，建立了 95% 置信区间，结果呈现在表1中。同时我们还对 $\hat{\mathbf{z}}$ 进行了可视化，如图5所示，其中零元素被标黑以示区分。由图可直观地看出剪枝后（右）编码中的零元素显著多于剪枝前（左）。

此外，表2表明 CAE-ADMM 在与传统方法相比较时仍能保证可以接受的运行速度（推断速度），注意如果进一步增大 batch size（并行化处理），CAE-ADMM 的平均推理速

度还可以进一步提升。

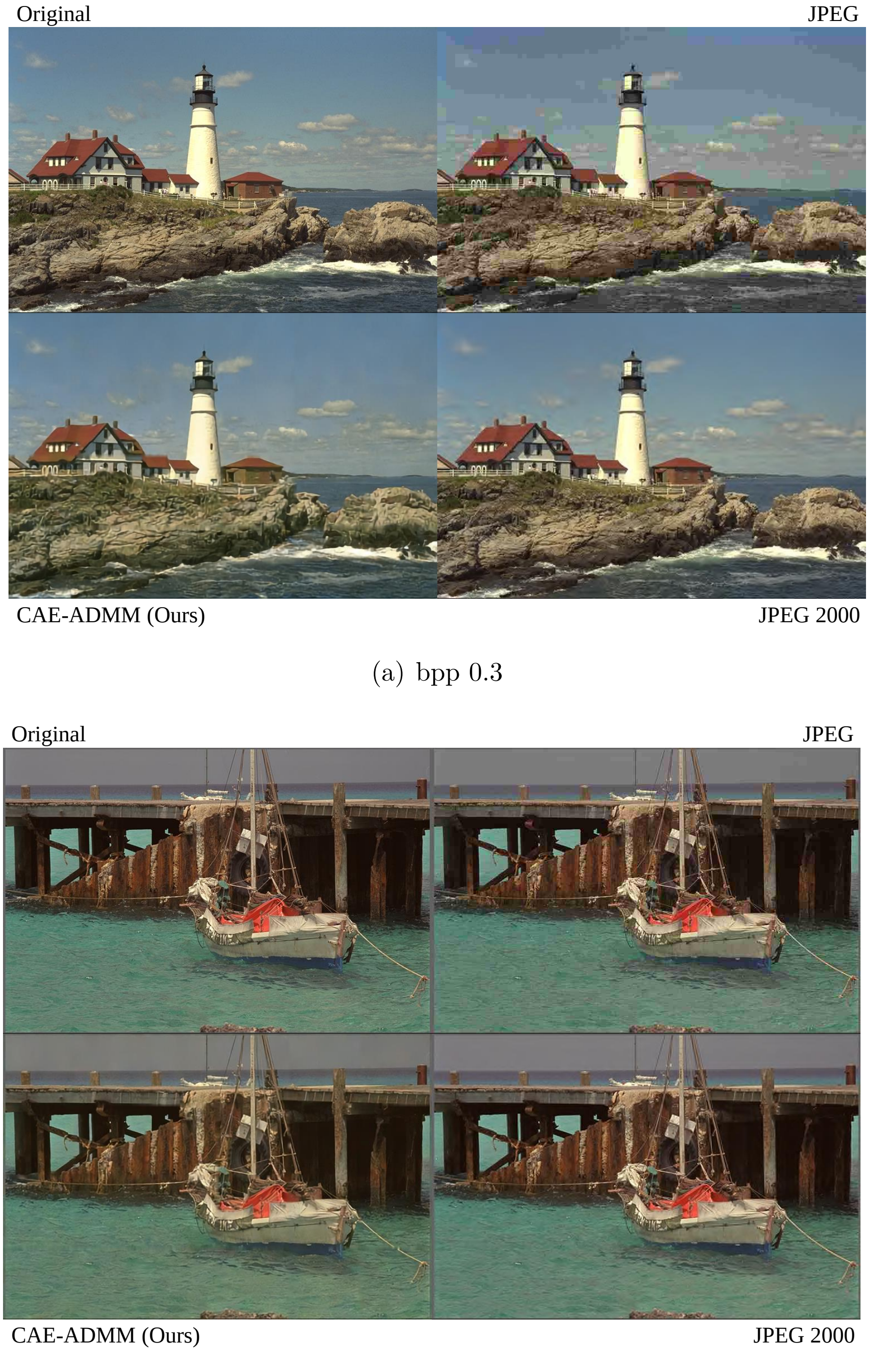
Model	$\overline{\text{bpp}}$	SSIM	Second/image
bpp_0.5	**0.597**	**0.871±0.003**	0.140±0.008
JPEG	0.603	0.828±0.006	**0.033±0.001**
JPEG2000	0.601	0.793±0.013	0.177±0.020

表 2: CAE-ADMM 的推理速度与传统方法的对比（采用混合测试集，2 个 GPU，batch size 设为 8，并建立了 95% 置信区间）。

5 结论

本研究提出了一种基于 ADMM 剪枝的隐式比特率优化方法，引入了该方法的 CAE-ADMM 模型，成功避开了额外训练熵编码器的繁琐与优化的不直接，对编码层剪枝，优化更为直接，更易实现且参数量更小。

本研究首次在图像压缩领域中引入剪枝方法，在熵估计的方法之外，探索了应用架构搜索方法的可能性。实验表明，CAE-ADMM 在 SSIM 和 MS-SSIM 下的表现均优于原本的 CAE 模型及其他现有的方法。通过 Ablation Study，我们论证了引入的 ADMM 剪枝方法的有效性。本文以简单的 ADMM 剪枝方法为例，为未来引入更多更精巧的架构搜索方法做了铺垫。



(a) bpp 0.3

(b) bpp 0.5

图 6: 不同方法的实际压缩效果：原图（左上）、CAE-ADMM（左下）、JPEG（右上）、JPEG 2000（右下）。

参考文献

[1] THEIS L, SHI W, CUNNINGHAM A, et al. Lossy image compression with compressive autoencoders[J]. arXiv preprint arXiv:1703.00395, 2017.

[2] WALLACE G K. The JPEG still picture compression standard[J]. IEEE transactions on consumer electronics, 1992, 38(1): xviii−xxxiv.

[3] TODERICI G, O’MALLEY S M, HWANG S J, et al. Variable rate image compression with recurrent neural networks[J]. arXiv preprint arXiv:1511.06085, 2015.

[4] TODERICI G, VINCENT D, JOHNSTON N, et al. Full Resolution Image Compression with Recurrent Neural Networks.[C] // CVPR. 2017: 5435−5443.

[5] MENTZER F, AGUSTSSON E, TSCHANNEN M, et al. Conditional probability models for deep image compression[C] // IEEE Conference on Computer Vision and Pattern Recognition (CVPR): Vol 1. 2018: 3.

[6] YE S, ZHANG T, ZHANG K, et al. Progressive Weight Pruning of Deep Neural Networks using ADMM[J]. arXiv preprint arXiv:1810.07378, 2018.

[7] ZHANG T, YE S, ZHANG K, et al. A Systematic DNN Weight Pruning Framework using Alternating Direction Method of Multipliers[J]. arXiv preprint arXiv:1804.03294, 2018.

[8] WANG Z, SIMONCELLI E P, BOVIK A C. Multiscale structural similarity for image quality assessment[C] // The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003: Vol 2. 2003: 1398−1402.

[9] WANG Z, BOVIK A C, SHEIKH H R, et al. Image quality assessment: from error visibility to structural similarity[J]. IEEE transactions on image processing, 2004, 13(4) : 600 – 612.

[10] JIANG F, TAO W, LIU S, et al. An end-to-end compression framework based on convolutional neural networks[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2017.

[11] BALLÉ J, LAPARRA V, SIMONCELLI E P. End-to-end optimization of nonlinear transform codes for perceptual quality[C] // Picture Coding Symposium (PCS), 2016. 2016 : 1 – 5.

[12] AGUSTSSON E, MENTZER F, TSCHANNEN M, et al. Soft-to-hard vector quantization for end-to-end learning compressible representations[C] // Advances in Neural Information Processing Systems. 2017 : 1141 – 1151.

[13] LI M, ZUO W, GU S, et al. Learning convolutional networks for content-weighted image compression[J]. arXiv preprint arXiv:1703.10553, 2017.

[14] RIPPEL O, BOURDEV L. Real-time adaptive image compression[J]. arXiv preprint arXiv:1705.05823, 2017.

[15] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network[C] // Advances in neural information processing systems. 2015 : 1135 – 1143.

[16] YANG T J, CHEN Y H, SZE V. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning[J], 2017 : 6071 – 6079.

[17] GUO Y, YAO A, CHEN Y. Dynamic Network Surgery for Efficient DNNs[J], 2016, to appear.

[18] DAI X, YIN H, JHA N K. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm[J]. arXiv preprint arXiv:1711.02017, 2017.

[19] WEN W, WU C, WANG Y, et al. Learning Structured Sparsity in Deep Neural Networks[J], 2016.

[20] HE Y, ZHANG X, SUN J. Channel Pruning for Accelerating Very Deep Neural Networks[J], 2017.

[21] ZOPH B, LE Q V. Neural Architecture Search with Reinforcement Learning[J], 2016.

[22] MILLER G F, TODD P M, HEGDE S U. Designing Neural Networks using Genetic Algorithms.[C] // The International Conference on Genetic Algorithms. 1989 : 379 – 384.

[23] LECUN Y, BENGIO Y, OTHERS. Convolutional networks for images, speech, and time series[J]. The handbook of brain theory and neural networks, 1995, 3361(10) : 1995.

[24] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[J], 2015 : 770 – 778.

[25] SHI W, CABALLERO J, HUSZAR F, et al. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network[J], 2016 : 1874 – 1883.

[26] ZEILER M D, FERGUS R. Visualizing and Understanding Convolutional Networks[J], 2013, 8689 : 818 – 833.

[27] BOYD S, PARIKH N, CHU E, et al. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers[J]. Foundations & Trends in Machine Learning, 2011, 3(1) : 1 – 122.

[28] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[29] MARTIN D, FOWLKES C, TAL D, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics[C] // Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on : Vol 2. 2001 : 416 – 423.

[30] LAI W-S, HUANG J-B, AHUJA N, et al. Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks[J]. arXiv:1710.01992, 2017.

6 附录：已预印版本 arXiv:1901.07196

本文所述研究成果已于 arXiv:1901.07196 预印，特附于此页后。

CAE-ADMM: IMPLICIT BITRATE OPTIMIZATION VIA ADMM-BASED PRUNING IN COMPRESSIVE AUTOENCODERS

Haimeng Zhao
Shanghai High School
Shanghai, China

Peiyuan Liao
Kent School*
Kent, CT, USA

ABSTRACT

We introduce ADMM-pruned Compressive AutoEncoder (CAE-ADMM) that uses Alternative Direction Method of Multipliers (ADMM) to optimize the trade-off between distortion and efficiency of lossy image compression. Specifically, ADMM in our method is to promote sparsity to implicitly optimize the bitrate, different from entropy estimators used in the previous research. The experiments on public datasets show that our method outperforms the original CAE and some traditional codecs in terms of SSIM/MS-SSIM metrics, at reasonable inference speed.

***Index Terms*—** autoencoder, lossy image compression, neural network pruning, bitrate optimization

1. INTRODUCTION

Since the proposal of compressive autoencoder (CAE) by [1], deep learning-approaches have been largely successful in the field of lossy image compression, where its adaptive feature learning capabilities have helped it outperform traditional codecs such as JPEG [2] and JPEG 2000 [3]. The goal of such network is to optimize the trade-off between the amount of distortion and the efficiency of the compression, usually expressed by the bitrate or bits per pixel (bpp). In other words, we aim to minimize

$$d(\mathbf{x}, \hat{\mathbf{x}}) + \beta \cdot R(\hat{\mathbf{z}}), \tag{1}$$

where d measures the distortion between the input image $\mathbf{x}$ and the reconstructed $\hat{\mathbf{x}}$, $\beta > 0$ controls the proportion, and R measures the bitrate of the quantized latent code $\hat{\mathbf{z}}$.

However, this objective is inherently non-differentiable due to the discrete nature of bitrate and quantization. So, to make the problem well-defined so that back-propagation of neural networks is applicable, one of the prevalent solutions proposed by [1] includes a combination of entropy coding and entropy rate estimation, in which they trained a parameterized entropy estimator $H : \mathbb{Z}^M \to [0, 1]$ along with the encoder/decoder that is further made differentiable by an upper-bounding process and the usage of Gaussian scale mixtures (GSMs).

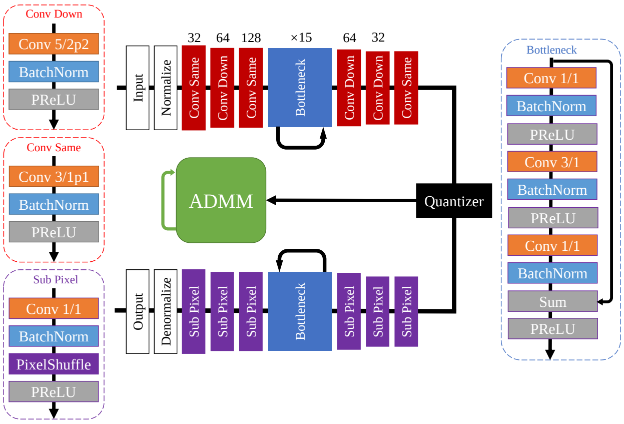


Fig. 1. The architecture of CAE-ADMM. “Conv k/spP” stands for a convolutional layer with kernel size k×k with a stride of s and a reflection padding of P, and “Conv Down” is reducing the height and weight by 2.

As an alternative to methods mentioned above, we replace the entropy estimator by an alternating direction method of multipliers module, where the aggressive pruning on the latent code encourages sparsity, and therefore aid in the optimization of the bitrate. Our experiments show that this pruning paradigm itself is capable of implicitly optimizing the entropy rate while yielding a better result compared with the original CAE and other traditional codecs when measured in both SSIM and MS-SSIM.

2. RELATED WORKS

In the literature, there exists numerous works on variants of compressive autoencoders (CAE) to achieve lossy image compression [1, 4, 5, 6, 7], with different approaches to measure the distortion and bitrate. For distance function $d(\mathbf{x}, \hat{\mathbf{x}})$, MSE (Mean-Squared Error), PSNR $= 10 \cdot \log_{10}(\frac{\text{MAX}_I^2}{\text{MSE}})$ (Peak Signal-to-noise Ratio), SSIM$(\mathbf{x}, \hat{\mathbf{x}}) =$

*Work done as a senior researcher at Kent Artificial Intelligence Laboratory.

$\frac{(2\mu_{\mathbf{x}}\mu_{\hat{\mathbf{x}}}+(k_1L)^2)(2\sigma_{\mathbf{x}\hat{\mathbf{x}}}+(k_2L)^2)}{(\mu_{\mathbf{x}}^2+\mu_{\hat{\mathbf{x}}}^2+(k_1L)^2)(\sigma_{\mathbf{x}}^2+\sigma_{\hat{\mathbf{x}}}^2+(k_2L)^2)}$ (Structural Similarity Index) and MS-SSIM (Multiscale SSIM) are the most widely used metrics. For the entropy estimator H, apart from the GSM model mentioned in the former section, recent works also use generative models[5] and context models[8] as potential alternatives. For works of non-autoencoder approaches, there is also an increasing interest in the usage of GAN [9], GDN (Generalized Divisive Normalization) [5] and RNN [10].

The traditional convex optimization ADMM algorithm [11] first saw its application in the field of neural architecture search (NAS) by the method proposed by Han et al. [12], which is followed by a few improvements in [13, 14, 15, 16].

3. PROPOSED METHOD

A typical CAE consists of an enocder E, a decoder D and a quantizer Q [1]:

$$\begin{aligned} E &: \mathbb{R}^n \to \mathbb{R}^m, \\ D &: \mathbb{R}^m \to \mathbb{R}^n, \\ Q &: \mathbb{R}^m \to \mathbb{Z}^m. \end{aligned}$$

The encoder E maps the original image $\mathbf{x} \in \mathbb{R}^n$ to a latent representation $\mathbf{z} = E(\mathbf{x})$. The quantizer Q then maps each element of $\mathbf{z}$ to $\mathbb{Z}$, which produces the compressed presentation of the image $\hat{\mathbf{z}} = Q(\mathbf{z})$. Finally, the decoder D attempts to reconstruct the original image $\hat{\mathbf{x}} = D(\hat{\mathbf{z}})$ from the information in $\hat{\mathbf{z}}$.

We aim to let the reconstructed image $\hat{\mathbf{x}}$ looks as similar as the original $\mathbf{x}$ (minimize the distance function d) while reducing the number of bits needed to store the latent code, or minimize the bitrate R. The problem can then be rephrased as below, assuming that Q is not parameterized:

$$\underset{E,D}{\operatorname{argmin}}\ d(\mathbf{x}, D \circ Q \circ E(\mathbf{x})) + \beta \cdot R \circ Q \circ E(\mathbf{x}) \tag{2}$$

3.1. Selection of E, D and Q

Similar to [1], CAE-ADMM uses convolutional layers to be the basis of our encoder and decoder. The decoder mirrors the structure of the encoder to maintain symmetry, except that uses sub-pixel convolutional layers proposed by Shi et al.[17] to perform up-sampling.

For the quantizer Q, we use a simple and computationally efficient one proposed by Theis et al.[1], inspired by the random binary version developed by Tordérici et al.[10]. It is defined as:

$$Q(t) = \lfloor t \rfloor + \epsilon,\ \epsilon \in \{0, 1\}, \tag{3}$$

in which ϵ decides whether to output the ground or the ceiling of the input, and the probability of $\epsilon = 1$ satisfies $P(\epsilon = 1) = t - \lfloor t \rfloor$. To make the quantizer differentiable, we define its gradient with that of its expectation:

$$\frac{\partial}{\partial t}Q(t) = \frac{\partial}{\partial t}\mathbb{E}[Q(t)] = \frac{\partial}{\partial t}t = 1 \tag{4}$$

3.2. Solution to the optimization problem

Since multiple well-defined metrics (mentioned in section 2) exist for d, we here aim to provide an alternative method to optimize R without the use of H. Intuitively, we can reformulate R by

$$R(\hat{\mathbf{z}}) = \text{card}(\hat{\mathbf{z}}) = \text{card}(Q \circ E(\mathbf{x})), \tag{5}$$

in which card$(\cdot)$ counts the number of non-zero elements. If we want z generated by the encoder to have fewer number of non-zero elements than a desired number ℓ, we can rephrase the problem into an ADMM-solvable problem [16]:

$$\begin{aligned} &\underset{E,D}{\operatorname{argmin}}\ d(\mathbf{x}, \hat{\mathbf{x}}) + g(\mathbf{Z}), \\ &\text{s.t. } Q \circ E(\mathbf{x}) - \mathbf{Z} = 0. \end{aligned} \tag{6}$$

where the indicator function $g(\cdot)$ is defined as

$$g(\mathbf{Z}) = \begin{cases} 0 & \text{if card}(\mathbf{Z}) \le \ell, \\ +\infty & \text{otherwise.} \end{cases} \tag{7}$$

Remark that both $\mathbf{U}$ and $\mathbf{Z}$ are initialized to be all-zero, and $\mathbf{Z}$ is an element of $\mathbf{S} = \{\mathbf{Z} |\ \text{card}(\mathbf{Z}) \le \ell\}$. By introducing the dual variable $\mathbf{U}$ and a penalty factor $\rho > 0$, we can split the above problem into two sub-problems. The first sub-problem is:

$$\underset{E,D}{\operatorname{argmin}}\ d(\mathbf{x}, \hat{\mathbf{x}}) + \frac{\rho}{2}\|Q \circ E(\mathbf{x}) - \mathbf{Z}^k + \mathbf{U}^k\|_F^2, \tag{8}$$

in which k is the current iteration number and $\|\cdot\|_F^2$ is the Frobenius norm. This is the neural network's loss with L_2 regularization, which can be solved by back propagation and gradient descent. The second sub-problem is:

$$\underset{Z}{\operatorname{argmin}}\ g(\mathbf{Z}) + \frac{\rho}{2}\|Q^{k+1} \circ E^{k+1}(\mathbf{x}) - \mathbf{Z} + \mathbf{U}^k\|_F^2. \tag{9}$$

The solution to this problem was derived by Boyd et al. in 2011[18]:

$$\mathbf{Z}^{k+1} = \Pi_{\mathbf{S}}(Q^{k+1} \circ E^{k+1}(\mathbf{x}) + \mathbf{U}^k), \tag{10}$$

where $\Pi_{\mathbf{S}}(\cdot)$ represents the Euclidean projection onto the set $\mathbf{S}$. Generally, Euclidean projection onto a non-convex set is difficult, but Boyd et al.[18] have proved that the optimal solution is to keep the ℓ largest elements of $Q^{k+1} \circ E^{k+1}(\mathbf{x}) + \mathbf{U}^k$ and set the rest to zero. Finally, we will update the dual variable $\mathbf{U}$ with the following policy:

$$\mathbf{U}^{k+1} = \mathbf{U}^k + Q^{k+1} \circ E^{k+1}(\mathbf{x}) - \mathbf{Z}^{k+1}. \tag{11}$$

These three steps together form one iteration of the ADMM pruning method. Algorithm 1 shows the complete steps.

Algorithm 1 Pruning of CAE Based on ADMM

```
Input:
  $\mathbf{x}$: A batch of input images;
  $E, D$: The encoder and decoder;
  $Q$: The quantizer;
  $\ell$: The expected number of non-zero elements in $E(\mathbf{x})$;
  $k_m$: Max number of iterations.
Output:
  $E, D$: Trained encoder and decoder;
  $\mathbf{U}, \mathbf{Z} \Leftarrow$ zeroes with the same shape as $Q \circ E(\mathbf{x})$
  for $1 \leq k \leq k_m$ do
      $E, D \Leftarrow \mathrm{argmin}_{E,D}\ d(\mathbf{x}, \hat{\mathbf{x}}) + \frac{\rho}{2}\|Q \circ E(\mathbf{x}) - \mathbf{Z} + \mathbf{U}\|_F^2$;
      $\mathbf{Z} \Leftarrow$ keep the $\ell$ largest elements in $Q \circ E(\mathbf{x}) + \mathbf{U}$ and set the rest to 0;
      $\mathbf{U} \Leftarrow \mathbf{U} + Q \circ E(\mathbf{x}) - \mathbf{Z}$.
  end for
  return $E, D$
```

4. EXPERIMENT

4.1. Model architecture

Our model architecture, shown in Fig. 1, is a modification of CAE proposed by [1]. The encoder and decoder are composed of convolutional layers as described in Section 3.1. The input image is first down-sampled by three blocks with each containing a convolutional layer, a batch normalization layer and a PReLU layer. Following 15 residual blocks, two more down-sampling convolutional blocks with the last convolutional block are applied, generating $\mathbf{z}$. The quantizer Q then quantizes it and fed into the decoder whose architecture mirrors the encoder.

4.2. Training

We use the Adam optimizer [19] with the batch size set to 32 to solve the first sub-problem. Learning rate is set to $4 \cdot 10^{-3}$ and is halved each time the loss has not dropped for ten epochs. Every 20 epochs, the second and third steps of the ADMM pruning method is applied. The distance function used as a part of back-propagation is a linear combination of MSE and differentiable versions of PSNR/SSIM/MS-SSIM, and the training is first warmed up by a scaled MSE alone. The ratio of the number of elements to retain in step two is set to be 10%. To enable fine-grained tuning of bpp, we modify the last layer of the encoder. All procedures are implemented in PyTorch and open-sourced[1]. Each model is trained for 300 epochs on 4 NVIDIA GeForce GTX 1080Ti GPUs.

4.3. Datasets and preprocessing

We use BSDS500 [20] as the training set, which contains five hundred 481×321 natural images. The images are randomly cropped to 128×128, horizontally and vertically flipped and then normalized. For the test set, we use the Kodak PhotoCD dataset [2], which contains twenty-four 768×512 images.

4.4. Results and discussion

We test CAE-ADMM (Our method), JPEG (implemented by libjpeg[3]) and JPEG 2000 (implemented by Kadadu Software[4]) on the Kodak PhotoCD dataset. For the distance metric, we use the open-source implementation of SSIM and MS-SSIM [5].

Fig. 2 shows a comparison of the performance achieved by the mentioned methods on Kodak. Our method (CAE-ADMM) outperforms all the other methods in both SSIM and MS-SSIM, especially the original CAE which uses entropy coding. Note that the blue curve represents the RNN-based method proposed by Toderici et al. which is optimized without an entropy estimator.

In Fig. 3, we demonstrate the effect of different compression methods visually: the origin (top left), JPEG (top right), CAE-ADMM (ours, bottom left) and JPEG 2000 (bottom right). From the figure, we can see that JPEG breaks down under a bpp of 0.3 while that of CAE-ADMM and JPEG 2000 are still satisfactory.

Model	bpp	ratio of zeros
Before pruning	1.684 ± 0.012	$7.80\% \pm 3.44\%$
After pruning	1.257 ± 0.011	$17.65\% \pm 4.90\%$

Table 1. Bpp & proportion of zero elements in $\hat{\mathbf{z}}$ the total number of elements in $\hat{\mathbf{z}}$ before and after pruning. For both statistics, a 95% confidence interval is established with a sample size of 233 (size of the mixed dataset).

For ablation study, we test out the effectiveness of ADMM-module by applying the same training procedure to the same model, one with the pruning schedule and another without. Then, we calculate the average bpp as well as the ratio of zero elements in a mixed dataset (768×512 crops of images from Urban100 [22], Manga109 [22] and Kodak PhotoCD). Results can be seen in Table 1 and more direct visualization of a sample image can be found in Figure 4.

Inference-speed-wise, from Table 2 we can see that our CAE-ADMM has an acceptable inference speed comparing to traditional codecs while maintaining superior quality concerning SSIM.

[1] https://github.com/JasonZHM/CAE-ADMM

[2] http://r0k.us/graphics/kodak/

[3] http://libjpeg.sourceforge.net/

[4] http://kakadusoftware.com/

[5] https://github.com/jorge-pessoa/pytorch-msssim

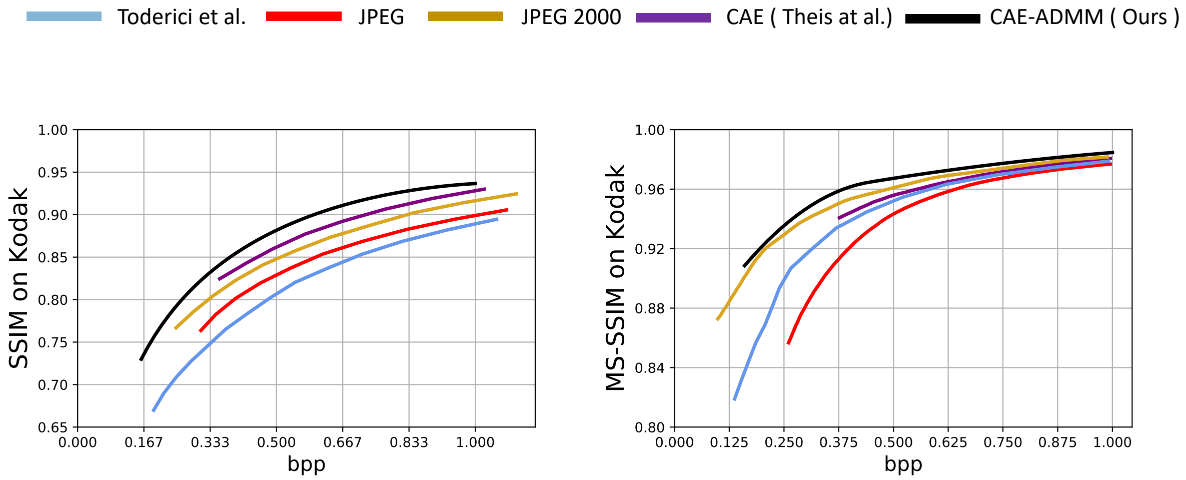


Fig. 2. Comparison of different method with respect to SSIM and MS-SSIM on the Kodak PhotoCD dataset. Note that Toderici et al.[21] used RNN structure instead of entropy coding while CAE-ADMM (Ours) replaces entropy coding with pruning method.

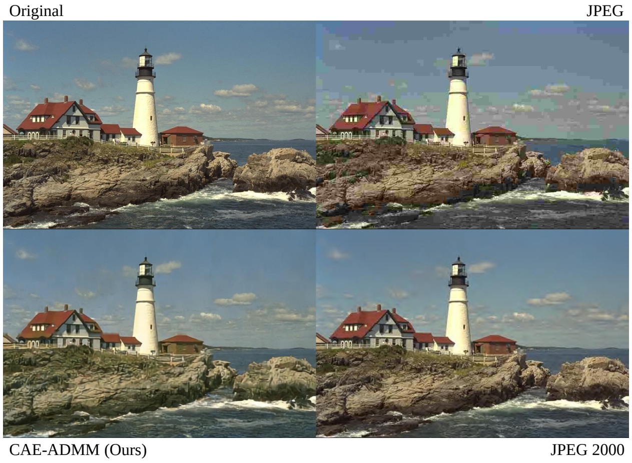


Fig. 3. Performance of different methods on image *kodim21* from Kodak dataset. Bpp is set to be about 0.3.

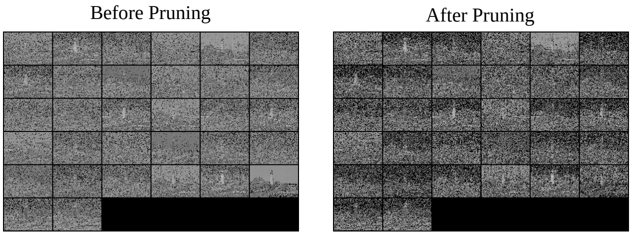


Fig. 4. Comparison of latent code before and after pruning for *kodim21*. For the sake of clarity, we marked zero values in the feature map before normalization as black.

Model	$\overline{\text{bpp}}$	SSIM	Second/image
bpp_0.5	**0.597**	**0.871±0.003**	0.140±0.008
JPEG	0.603	0.828±0.006	**0.033±0.001**
JPEG2000	0.601	0.793±0.013	0.177±0.020

Table 2. 95% confidence intervals of SSIM and inference speed of CAE-ADMM model (inference time being the sum of all 128 × 128 patches with batch size = 8) on mixed dataset, 2 GPUs compared to traditional codecs.

5. CONCLUSION

In this paper, we propose the compressive autoencoder with ADMM-based pruning (CAE-ADMM) [23], which serves as an alternative to the traditionally used entropy estimating technique for deep-learning-based lossy image compression. Tests on multiple datasets show better results than the original CAE model along with other contemporary approaches. We further explore the effectiveness of the ADMM-based pruning method by looking into the detail of learned latent codes.

Further study can focus on developing a more efficient pruning method, e.g., introducing ideas from the field of reinforcement learning. Also, the structures of E, D, and Q can be further optimized for speed and accuracy.

6. REFERENCES

[1] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár, “Lossy image compression with compressive autoencoders,” *arXiv preprint arXiv:1703.00395*, 2017.

[2] Gregory K Wallace, “The jpeg still picture compression standard,” *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[3] David S. Taubman and Michael W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*, 2002.

[4] Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao, “An end-to-end compression framework based on convolutional neural networks,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[5] Johannes Ballé, Valero Laparra, and Eero P Simoncelli, “End-to-end optimization of nonlinear transform codes for perceptual quality,” in *Picture Coding Symposium (PCS), 2016*. IEEE, 2016, pp. 1–5.

[6] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool, “Soft-to-hard vector quantization for end-to-end learning compressible representations,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1141–1151.

[7] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang, “Learning convolutional networks for content-weighted image compression,” *arXiv preprint arXiv:1703.10553*, 2017.

[8] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool, “Conditional probability models for deep image compression,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, vol. 1, p. 3.

[9] Oren Rippel and Lubomir Bourdev, “Real-time adaptive image compression,” *arXiv preprint arXiv:1705.05823*, 2017.

[10] George Toderici, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar, “Variable rate image compression with recurrent neural networks,” *arXiv preprint arXiv:1511.06085*, 2015.

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[12] Song Han, Jeff Pool, John Tran, and William Dally, “Learning both weights and connections for efficient neural network,” in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[13] Tien Ju Yang, Yu Hsin Chen, and Vivienne Sze, “Designing energy-efficient convolutional neural networks using energy-aware pruning,” pp. 6071–6079, 2017.

[14] Yiwen Guo, Anbang Yao, and Yurong Chen, “Dynamic network surgery for efficient dnns,” vol. to appear, 2016.

[15] Yihui He, Xiangyu Zhang, and Jian Sun, “Channel pruning for accelerating very deep neural networks,” 2017.

[16] Shaokai Ye, Tianyun Zhang, Kaiqi Zhang, Jiayu Li, Kaidi Xu, Yunfei Yang, Fuxun Yu, Jian Tang, Makan Fardad, Sijia Liu, et al., “Progressive weight pruning of deep neural networks using admm,” *arXiv preprint arXiv:1810.07378*, 2018.

[17] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” pp. 1874–1883, 2016.

[18] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations & Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[19] Diederik P Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[20] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. IEEE, 2001, vol. 2, pp. 416–423.

[21] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell, “Full resolution image compression with recurrent neural networks.,” in *CVPR*, 2017, pp. 5435–5443.

[22] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang, “Fast and accurate image super-resolution with deep laplacian pyramid networks,” *arXiv:1710.01992*, 2017.

[23] Haimeng Zhao and Peiyuan Liao, “Cae-admm: Implicit bitrate optimization via admm-based pruning in compressive autoencoders,” *arXiv:1901.07196*, 2019.

7 致谢

本研究的选题来自最新计算机视觉方向的顶级会议论文（ICLR 2017、CVPR 2018 等），是我在阅读 Theis et al. 的 [1]“Lossy image compression with compressive autoencoders”及 Ye et al. 的 [2]“Progressive weight pruning of deep neural networks using admm”这两篇论文时想到的。主要的启发来自于 [2]，一篇应用剪枝技术进行神经网络架构搜索的研究，看到这篇论文时我刚好读过 [1]，想起来了 [1] 中提到的一个难点，我就想能不能用 [2] 的方法来改进 [1]，没想到实验效果还不错。

本研究的论文撰写、理论推演、代码编写、实验设计与实施、结果的分析与比较等工作的全部内容均由我本人独立完成。

我的辅导老师是学校的计算机老师。我和另外三名同学都是计算机课题小组的成员，每周二、周四下午会抽出 3 到 4 小时的时间集中到一起专门进行计算机课题的研究。一开始老师会请来自复旦、交大的教授来做讲座，后来的时间主要是我们各自进行自己的研究，老师负责场地、时间、后勤等安排和协调。课题指导是学校开展的课余活动，对于我们学生而言没有支付任何费用。

特别感谢上海交通大学学生创新中心高性能计算平台楚朋志老师提供的计算资源支持。在将论文由中文翻译至英文的过程中，美国高中的朋友 Peiyuan Liao 给予了我很大的帮助，他同时也在研究后期参与了一些实验的完善与补充，因而成为了我的论文的第二作者。同时本研究中大量应用了开源社区 GitHub 中的成果，因而本文的成果也已经在 GitHub 开源。

本课题的研究过程是很艰苦的，由于研究内容是学界今年来迭代迅速的前沿领域，进行研究所需要的知识储备跨度极大，需要投入很多的时间和精力，这对一个同时还需要准备高考和学科竞赛的高中生而言是一个极大的挑战。很幸运的是，我完成了这个研究，并以此在上海市青少年科技创新大赛、上海市明日科技之星等活动中取得了不错的名次。这样的成果要归功于一直以来给予我无条件支持与陪伴的父母、老师，给我提供了学习和研究平台的学校，以及无数在与我交流过程中给我启发和帮助的朋友。在此对他们表达诚挚的感谢。

此页为学术诚信声明

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员： 赵海萌　　　**指导老师：** 沈孝山

2019 年 8 月 26 日