AutoMC: 基于领域知识和渐进式搜索的自动模型压缩*

1190200523 石翔宇

日期：2022 年 6 月 4 日

摘　要

模型压缩可以在保持性能的前提下降低神经网络模型的复杂度，从而促进深度神经网络在资源受限环境下的应用。尽管它们取得了巨大的成功，但选择一个合适的压缩方法和设计压缩方案的细节都很困难，需要大量的领域知识作为支持，这对非专业用户并不友好。为了让更多的用户轻松获得最符合他们需求的模型压缩方案，在本文中，我们提出了 AutoMC，一种有效的模型压缩自动化工具。AutoMC 构建了模型压缩领域知识，深入了解每种压缩方法在不同设置下的特点和优势。我们提出了一种渐进式搜索策略，根据学习到的先验知识结合历史评价信息，有效探索帕累托最优压缩方案。大量的实验结果表明，AutoMC 可以在短时间内提供令人满意的压缩方案，证明了 AutoMC 的有效性。

1　引言

神经网络非常强大，可以处理许多现实世界的任务，但它们的参数量通常非常大，带来昂贵的计算和存储成本。为了将它们应用到移动设备中，已经提出了许多模型压缩方法，包括模型剪枝 [2, 6, 9, 16, 23]、知识蒸馏 [28]、低秩分解近似 [2, 15] 等等。

这些压缩方法可以在尽可能保持模型精度的同时有效地减少模型参数，但是使用起来比较困难。每种方法都有很多超参数会影响其压缩效果，而且不同的方法适合的模型压缩任务也不同。即使是领域专家也需要大量时间来测试和分析，以便为给定的压缩任务设计合理的压缩方案。这给压缩技术的实际应用带来了很大的挑战。

为了使普通用户能够轻松有效地使用现有的模型压缩技术，在本文中，我们提出了 AutoMC，一种自动化机器学习算法（AutoML）来帮助用户自动设计模型压缩方案。请注意，在 AutoMC 中，我们不会将压缩方案限制为仅在特定设置下使用压缩方法。相反，我们允许不同的压缩方法和不同超参数设置下的方法一起工作（顺序执行）以获得多样化的压缩方案。我们试图通过这种顺序组合来整合不同方法和设置的优势，从而获得更强大的压缩效果，我们最终的实验结果证明了这个想法是有效和可行的。

但是，AutoMC 的搜索空间巨大。压缩方案中包含的压缩策略的数量[1]可能是任意大小，这给后续的搜索任务带来了很大的挑战。为了提高搜索效率，我们分别从知识引入和搜索空间缩减的角度提出了以下两项创新来提高 AutoMC 的性能。

具体来说，对于第一个创新，我们构建了模型压缩领域的知识，揭示了压缩策略的技术和设置细节，以及它们在一些常见压缩任务下的性能。这些领域知识可以帮助 AutoMC 深入了解每个组件在搜索空间中的潜在特征和优势。它可以指导 AutoMC 选择更合适的压缩策略来构建有效的压缩方案，从而减少无用的评估，提高搜索效率。

第二个创新，我们采用了渐进式搜索空间扩展的思路，提高了 AutoMC 的搜索效率。具体来说，在每一轮优化中，我们只将评估的压缩方案的下一步操作，即未探索的下一步压缩策略作为搜索空间。然

*这是作者在本校海量数据计算研究中心实习时参与的工作，作者的主要贡献是论文中相关工作和实验部分的撰写，补充材料中算法收敛性的证明，实验代码的编写以及实验的设计及运行。

[1]在本文中，压缩策略是指具有特定超参数设置的压缩方法。

后，我们选择帕累托最优操作进行方案评估，最后将新方案的下一步操作作为新扩展的搜索区域参与下一轮优化。这样，AutoMC 可以选择性地、逐步地探索更有价值的搜索空间，降低搜索难度，提高搜索效率。此外，AutoMC 可以细粒度地分析比较后续操作对各个压缩方案性能的影响，最终确定更有价值的下一步探索路线用于实施，从而有效减少对无用方案的评估。

最终的实验结果表明 AutoMC 可以快速搜索到强大的模型压缩方案。与现有的非渐进式、忽略领域知识的 AutoML 算法相比，AutoMC 更适合处理搜索空间巨大，组件完整且可执行的算法的自动模型压缩问题。

2 相关工作

2.1 模型压缩算法

模型压缩是将神经网络应用于移动设备或嵌入设备的关键点，已在世界范围内得到广泛研究。研究人员提出了许多有效的压缩方法，大致可分为以下四类。(1) 剪枝方法，旨在从神经网络中去除冗余部分，例如过滤器、通道、内核或层 [8, 18, 19, 22]；(2) 知识蒸馏方法，在训练有素的大型模型的监督下训练紧凑且计算效率高的神经模型；(3) 使用分解技术将卷积矩阵分成小矩阵的低秩逼近方法 [17]；(4) 降低神经网络参数值精度的量化方法 [11, 30]。

这些压缩方法各有优势，在许多压缩任务中都取得了巨大的成功，但如引言部分所述，应用起来很困难。在本文中，我们旨在灵活地利用他们提供的经验来支持模型压缩方案的自动设计。

2.2 自动机器学习算法

自动机器学习算法的目标是实现机器学习的渐进式自动化，包括神经网络架构的自动化设计，机器学习工作流程 [10, 29] 和机器学习模型的超参数自动化搜索 [12, 24]。现有 AutoML 算法的思想是定义一个包含多种解的有效搜索空间，然后设计一种高效的搜索策略，从搜索空间中快速找到最佳机器学习解决方案，最后将最佳解决方案作为最终输出。

搜索策略对 AutoML 算法的性能影响很大。现有的 AutoML 搜索策略可以分为 3 类：基于强化学习（RL）的方法 [1]，基于遗传（EA）的方法 [4, 26] 和基于梯度的方法 [21, 25]。基于强化学习的方法使用循环网络作为控制器来确定一系列算子，从而依次构建机器学习解决方案。基于遗传的方法首先初始化一组机器学习解决方案，然后将它们的验证准确度作为适应度进行演化。至于基于梯度的方法，它们是为神经架构搜索问题而设计的。它们将搜索空间放宽为连续的，这样架构就可以通过梯度下降来优化其验证性能 [3]。它们无法处理由可执行压缩策略组成的搜索空间。因此，我们仅将 AutoMC 与前两种方法进行比较。

3 AutoMC 算法

我们首先给出模型压缩的相关概念和模型自动压缩的问题定义（节 3.1）。然后，我们充分利用已有的经验，为压缩算法构建一个高效的搜索空间（节 3.2）。最后，我们设计了一种搜索策略，从知识引入和搜索空间缩减的角度提高了搜索效率，帮助用户快速搜索到最优压缩方案（节 3.3）。

3.1 相关概念和问题定义

相关概念. 给定一个神经网络模型 M，我们使用 $P(M)$、$F(M)$ 和 $A(M)$ 分别表示它的参数量、FLOPS 和它在给定数据集上的准确率。给定模型压缩方案 $S = \{s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_k\}$，其中 s_i 是一种压缩

表 1: 在我们的搜索空间中使用的六种开源模型压缩方法。

标号	压缩算法	方法	超参数
C1	LMA [28]	TE_1: 基于 LMA 函数的知识蒸馏	• HP_1 : 微调轮数 • HP_2 : 参数量下降率 • HP_3 : LMA 的分段数量 • HP_4 : 温度系数 • HP_5 : alpha 因子
C2	LeGR [6]	TE_2: 基于遗传算法的过滤器剪枝 TE_3: 微调	• $HP1, HP2$: 和 C1 一样 • HP_6 : 通道的最大剪枝率 • HP_7 : 进化轮数 • HP_8 : 过滤器的评价标准
C3	NS [23]	TE_4: 基于 BN 层中缩放因子的通道修剪 TE_3: 微调	• HP_1, HP_2 : 和 C1 一样 • HP_6 : 和 C2 一样
C4	SFP [9]	TE_5: 基于反向传播的过滤器剪枝	• HP_2 : 和 C1 一样 • HP_9 : 反向传播轮数 • HP_{10} : 更新频率
C5	HOS [2]	TE_6: 基于 HOS 的过滤器剪枝 [27] TE_7: 基于 HOOI 的低秩核分解 [13] TE_3: 微调	• HP_1, HP_2 : 和 C1 一样 • HP_{11} : 全局评价标准 • HP_{12} : 全局评价标准 • HP_{13} : 优化轮数 • HP_{14} : MSE 损失因子
C6	LFB [15]	TE_9: 基于滤波器基的低秩滤波器分解	• HP_1, HP_2 : 和 C1 一样 • HP_{15} : 辅助 MSE 损失因子 • HP_{16} : 辅助损失

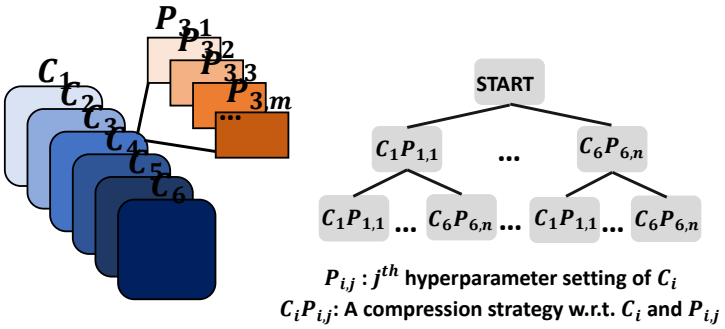


图 1: AutoMC 的搜索空间可以用树形结构来描述。每个节点有 4525 个子节点，对应 Table 1中的 4525 个压缩策略。

策略（压缩策略按顺序执行），我们用 $S[M]$ 表示将 S 应用到 M 后得到的压缩模型。另外，我们使用 $*R(S,M) = \frac{*(M)-*(S[M])}{*(M)} \in [0,1]$，其中 $*$ 可以为 P 或 F，表示模型 M 在执行 S 后对参数量或 FLOPS 的减少率。我们使用 $AR(S,M) = \frac{A(S[M])-A(M)}{A(M)} > -1$ 来表示 S 在 M 上实现的准确率提高率。

定义 1（模型自动压缩）给定一个神经模型 M、参数的目标缩减率 γ 和压缩方案上的搜索空间 $\mathbb{S}$，模型自动压缩问题旨在快速找到 $S^* \in \mathbb{S}$:

$$S^* = \underset{S\in\mathbb{S}, PR(S,M) geq \gamma}{\operatorname{argmax}} f(S,M)$$
$$f(S,M) := [AR(S,M), PR(S,M)] \tag{1}$$

一种帕累托最优压缩方案，在两个优化目标上表现良好：PR 和 AR，并且满足参数的目标缩减率。

3.2 压缩方案的搜索空间

在 AutoMC 中，我们利用一些开源的模型压缩方法来构建模型压缩的搜索空间。具体来说，我们收集了 6 种模型压缩方法，可以灵活地组合它们以获得多样化的模型压缩方案，以应对不同的压缩任务。另外，考虑到超参数对每种方法的性能影响很大，我们将不同超参数设置下的压缩方法视为不同的压缩策

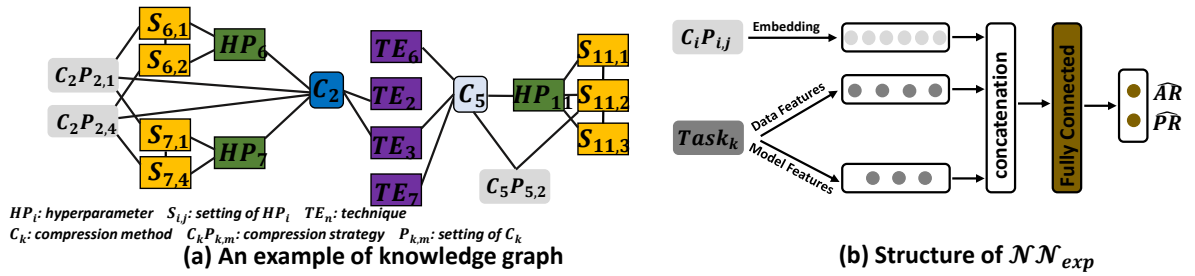


图 2: 用于嵌入学习的知识图谱和 $\mathcal{NN}_{exp}$ 的结构。$S_{i,j}$ 是超参数 HP_i 的设置。

略，并计划找到最佳的压缩策略序列，即压缩方案，以有效解决实际的压缩问题。

表 1 给出了这些压缩方法。这些方法及其各自的超参数构成了总计 $4,525$ 个压缩策略。利用这些压缩策略形成不同长度的压缩策略序列（长度 $< L$），那么我们得到一个搜索空间 $\mathbb{S}$，其中有 $\sum_{l=0}^{L}(4525)^l$ 种不同的压缩方案。

我们的搜索空间 $\mathbb{S}$ 可以描述为一个树形结构（如图 1），其中每个节点有 $4,525$ 个子节点，对应了 $4,525$ 种压缩策略。在这种树结构中，从 $START$ 节点到树中任意节点的每条路径都对应一个压缩策略序列，即搜索空间中的一个压缩方案。

3.3 AutoMC 算法的搜索策略

搜索空间 $\mathbb{S}$ 是巨大的。为了提高搜索性能，我们引入了领域知识来帮助 AutoMC 学习 $\mathbb{S}$ 中组件的特征（节 3.3.1）。此外，我们设计了一种渐进式搜索策略来精细分析后续操作对压缩方案的影响，从而提高搜索效率（节 3.3.2）。

3.3.1 基于领域知识的嵌入学习

我们构建了一个关于压缩策略的知识图谱，并从相关研究论文的实验中提取经验，以了解每种压缩策略在搜索空间中的潜在优势和有效嵌入。考虑到两种知识属于不同类型[2]，适用于不同的分析方法，我们为它们设计了不同的嵌入学习方法，结合两种方法以更好地理解不同的压缩策略。

基于知识图谱的嵌入学习. 我们构建了一个知识图 $\mathbb{G}$，其中揭示了每个压缩策略的技术和设置细节，以帮助 AutoMC 学习不同压缩策略之间的关系和差异。$\mathbb{G}$ 包含五种实体节点：(E_1) 压缩策略、(E_2) 压缩方法、(E_3) 超参数、(E_4) 超参数设置和 (E_5) 压缩技术。此外，它包括五种类型的实体关系：

R_1: 压缩策略与其压缩方法的对应关系（$E_1 \rightarrow E_2$）
R_2: 压缩策略与其超参数设置的对应关系（$E_1 \rightarrow E_4$）
R_3: 压缩方法与其超参数的对应关系（$E_2 \rightarrow E_3$）
R_4: 压缩方法与其压缩技术的对应关系（$E_2 \rightarrow E_5$）
R_5: 超参数与其设置的对应关系（$E_3 \rightarrow E_4$）

R_1 和 R_2 描述了压缩策略的组成细节，R_3 和 R_4 提供了压缩方法的简要描述，R_5 说明了超参数设置的含义。图 2 (a) 是 $\mathbb{G}$ 的一个例子。

我们使用 TransR [20] 将 $\mathbb{G}$ 中的实体和关系参数化为向量表示，同时保留 $\mathbb{G}$ 的图结构。具体来说，给定 $\mathbb{G}$ 中的三元组 (h, r, t)，我们通过优化变换原则来学习每个实体和关系的嵌入：

$$W_r e_h + e_r \approx W_r e_t \tag{2}$$

其中 $e_h e_t \in R^d$ 和 $e_r \in R^k$ 分别是 h、t 和 r 的嵌入; $W_r \in R^{k \times d}$ 是关系 r 的变换矩阵。

[2]知识图谱是关系知识而实验经验属于数值知识

Algorithm 1 压缩策略嵌入学习

$\mathbb{C} \leftarrow$ 表 1中的压缩策略
$\mathbb{G} \leftarrow$ 在 $\mathbb{C}$ 上构建知识图谱
$\mathbb{E} \leftarrow$ 从表 1 涉及的论文中提取 $\mathbb{G}$
while 轮数 $< TrainEpoch$ **do**
  使用 $\mathbb{G}$ 中的三元组执行 TransR 的一轮训练
  $e_{C_iP_{i,j}} \leftarrow$ 提取压缩策略的知识嵌入 $C_iP_{i,j}$ $(\forall C_iP_{i,j} \in \mathbb{C})$
  使用 $\mathbb{E}$ 根据公式 3 优化获得的知识嵌入
  $\widetilde{e}_{C_iP_{i,j}} \leftarrow$ 提取 $C_iP_{i,j}$ $(\forall C_iP_{i,j} \in \mathbb{C})$ 的增强嵌入
  将 $e_{C_iP_{i,j}}$ 替换为 $\widetilde{e}_{C_iP_{i,j}}$ $(\forall C_iP_{i,j} \in \mathbb{C})$
end while
return 压缩策略的高级嵌入: $\widetilde{e}_{C_iP_{i,j}}$ $(\forall C_iP_{i,j} \in \mathbb{C})$

这种嵌入学习方法可以将 $\mathbb{G}$ 中的知识注入到压缩策略的嵌入中，从而学习到压缩策略的有效表示。在 AutoMC 中，我们用 $e_{C_iP_{i,j}}$ 表示从 $\mathbb{G}$ 学习的压缩策略 $C_iP_{i,j}$ 的嵌入。

基于实验经验的嵌入增强. 研究论文包含许多有价值的实验经验：压缩策略在各种压缩任务下的性能，这些经验有助于深入理解每种压缩策略的性能特点。如果我们可以将它们整合到压缩策略的嵌入中，那么 AutoMC 可以在更高质量嵌入的指导下做出更准确的决策。

基于这个思想，我们设计了一个神经网络，记为 $\mathcal{NN}_{exp}$（如图 2 (b) 所示），进一步优化从 $\mathbb{G}$ 学习到的压缩策略的嵌入表示。$\mathcal{NN}_{exp}$ 将 $e_{C_iP_{i,j}}$ 和压缩任务 $Task_k$ 的特征向量（记为 e_{Task_k}）作为输入，打算输出 $C_iP_{i,j}$ 的压缩性能，包括参数的减少率 PR 和准确率的增加率 AR .

这里，$Task_k$ 由数据集属性和模型性能信息组成。以图像分类模型的压缩任务为例，特征向量可以由以下 7 部分组成：(1) 数据特征：类别数、图像大小、图像通道数和数据量。(2) 模型特征：原始模型的参数量、FLOPs 和数据集上的准确率。

在 AutoMC 中，我们从相关压缩论文中提取实验经验：$(C_iP_{i,j}, Task_k, AR, PR)$，然后输入 $e_{C_iP_{i,j}}$ 和 e_{Task_k} 到 $\mathcal{NN}_{exp}$ 以获得预测的性能分数，记为 $(\hat{AR}, \hat{PR})$。最后，我们优化 $e_{C_iP_{i,j}}$，得到 $C_iP_{i,j}$ 的更有效嵌入，记为 $\widetilde{e}_{C_iP_{i,j}}$，通过最小化 (AR, PR) 和 $(\hat{AR}, \hat{PR})$ 之间的差异：

$$\min_{\theta, e_{C_iP_{i,j}}(C_iP_{i,j}\in\mathbb{C})} \frac{1}{|\mathbb{E}|} \sum_{(C_iP_{i,j}, Task_k, AR, PR)\in\mathbb{E}} \|\mathcal{NN}_{exp}(e_{C_iP_{i,j}}, Task_k; \theta) - (AR, PR)\| \tag{3}$$

其中 θ 表示 $\mathcal{NN}_{exp}$ 的参数，$\mathbb{C}$ 表示表 1中的压缩策略集合，$\mathbb{E}$ 是从论文中提取的一组实验经验。

伪代码. 结合以上两种学习方法，AutoMC 可以综合考虑知识图谱和实验经验，获得更有效的嵌入。Algorithm 1给出了 AutoMC 嵌入学习部分的完整伪代码。

3.3.2 渐进式搜索策略

在搜索阶段以压缩方案为单位进行分析和评估可能非常低效，因为当其序列较长时，压缩方案的评估可能非常昂贵。搜索策略可能会花费大量时间进行评估，而只能获得较少的优化性能信息，这是非常低效的。

为了提高搜索效率，我们在 AutoMC 中应用了渐进式搜索策略的思想。我们尝试通过分析丰富的过程信息，即每种压缩策略对原始压缩策略序列的影响，逐步将有价值的压缩策略添加到评估的压缩方案中，以便从巨大的搜索空间中快速找到更好的方案 $\mathbb{S}$。

具体来说，我们利用历史信息来学习多目标评估器 $\mathcal{F}_{mo}$（如图 3所示）。我们使用 $\mathcal{F}_{mo}$ 来分析新加入的压缩策略 $s_{t+1} = C_iP_{i,j} \in \mathbb{C}$ 对压缩方案 $seq = (s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_t)$ 的性能，包括准确率提升率 AR_{step} 和参数缩减率 PR_{step}。

对于每一轮优化，我们首先对一些帕累托最优和评估方案 $seq \in \mathcal{H}_{scheme}$ 进行采样，并使用它们的

Algorithm 2 渐进式搜索策略

$\mathcal{H}_{scheme} \leftarrow \{START\}, OPT_{START} \leftarrow \mathbb{C}$
while epoch $<$ $SearchEpoch$ **do**
  $\mathcal{H}^{sub}_{scheme} \leftarrow$ 从 $\mathcal{H}_{scheme}$ 中采样一些方案
  $\mathbb{S}_{step} \leftarrow \{(seq, s) \mid \forall seq \in \mathcal{H}^{sub}_{scheme},\ s \in Next_{seq}\}$
  $ParetoO \leftarrow \text{argmax}_{(seq,s)\in\mathbb{S}_{step}}[ACC_{seq,s}, PAR_{seq,s}]$
  评估 $ParetoO$ 中的方案并得到 $AR^{seq^*,s^*}_{step}, PR^{seq^*,s^*}_{step}$ $\left((seq^*, s^*) \in ParetoO\right)$
  根据公式 5 优化多目标评估器 $\mathcal{F}_{mo}$ 的权重 ω
  $\mathcal{H}_{scheme} \leftarrow \mathcal{H}_{scheme} \cup \{seq^*, s^* \mid (seq^*, s^*) \in ParetoO)\}$
  $OPT_{seq^*} \leftarrow OPT_{seq^*} - \{s^*\}, OPT_{seq^* \leftarrow s^*} \leftarrow \mathbb{C}$ for each $(seq^*, s^*) \in ParetoO$
  $ParetoSchemes \leftarrow \mathcal{H}_{scheme}$ 中参数下降率 $\geq \gamma$ 的帕累托最优压缩方案
end while
return $ParetoSchemes$

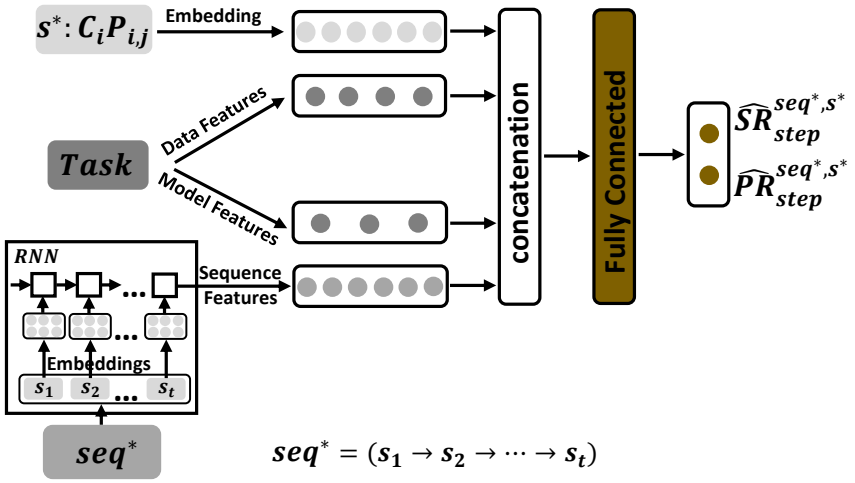


图 3: Structure of $\mathcal{F}_{mo}$. The embedding of s_i and s^* are provided by Algorithm 1.

下一步压缩策略 $Next_{seq} \subseteq \mathbb{C}$ 作为搜索空间 $\mathbb{S}_{step}$：$\mathbb{S}_{step} = \{(seq, s) | \forall seq \in \mathcal{H}^{sub}_{scheme}, s \in Next_{seq}\}$，其中 $\mathcal{H}^{sub}_{scheme} \subseteq \mathcal{H}_{scheme}$ 是取样的方案。其次，使用 $\mathcal{F}_{mo}$ 从 $\mathbb{S}_{step}$ 中选择帕累托最优 $ParetoO$，从而得到更好的压缩方案 $seq^* \rightarrow s^*, \forall (seq^*, s^*) \in ParetoO$ 来进行评估。

$$
\begin{aligned}
ParetoO &= \underset{(seq,s)\in\mathbb{S}_{step}}{\text{argmax}}\ [ACC_{seq,s}, PAR_{seq,s}] \\
ACC_{seq,s} &= A(seq[M]) \times (1 + \hat{AR}^{seq,s}_{step}) \\
PAR_{seq,s} &= P(seq[M]) \times (1 - \hat{PR}^{seq,s}_{step})
\end{aligned}
\tag{4}
$$

其中 $\hat{AR}^{seq,s}_{step}$ 和 $\hat{PR}^{seq,s}_{step}$ 是 $\mathcal{F}_{mo}$ 预测的 s 为方案 seq 带来的性能变化。$ACC_{seq,s}$ 和 $PAR_{seq,s}$ 是对原模型 M 执行方案 $seq \rightarrow s$ 后得到的准确率和参数量。

最后，我们评估 $ParetoO$ 中的压缩方案并得到它们的真实性能变化 $AR^{seq^*,s^*}_{step}, PR^{seq^*,s^*}_{step}$，并使用以下公式进一步优化 $\mathcal{F}_{mo}$ 的性能：

$$
\min_{\omega} \frac{1}{|ParetoO|} \sum_{(seq^*,s^*)\in ParetoO} \|\mathcal{F}_{mo}(seq^*, s^*; \omega) - (AR^{seq^*,s^*}_{step}, PR^{seq^*,s^*}_{step})\| \tag{5}
$$

我们将新方案 $\{seq^* \rightarrow s^* | (seq^*, s^*) \in ParetoO)\}$ 添加到 $\mathcal{H}_{scheme}$ 中以参与下一轮优化步骤。

渐进式搜索和 AutoMC 的优势。通过这样的方法，AutoMC 可以获得更多的训练数据进行策略优化，并且可以选择性地探索更多有价值的搜索空间，从而提高搜索效率。

将 Algorithm 1 学习到的嵌入应用到 Algorithm 2，即将学习到的压缩策略和先前的策略序列的高级嵌入输入到 $\mathcal{F}_{mo}$，然后我们就得到了 AutoMC。

表 2: 使用 ResNet-56 在 CIFAR-10 和 VGG-16 在 CIFAR-100 上的压缩结果。

PR(%)	Algorithm	ResNet-56 on CIFAR-10			VGG-16 on CIFAR-100		
		Params(M) / PR(%)	FLOPs(G) / FR(%)	Acc. / Inc.(%)	Params(M) / PR(%)	FLOPs(G) / FR(%)	Acc. / Inc.(%)
	baseline	0.90 / 0	0.27 / 0	91.04 / 0	14.77 / 0	0.63 / 0	70.03 / 0
≈ 40	LMA	0.53 / 41.74	0.15 / 42.93	79.61 / -12.56	8.85 / 40.11	0.38 / 40.26	42.11 / -39.87
	LeGR	0.54 / 40.02	0.20 / 25.76	90.69 / -0.38	8.87 / 39.99	0.56 / 11.55	69.97 / -0.08
	NS	0.54 / 40.02	0.12 / 55.68	89.19 / -2.03	8.87 / 40.00	0.42 / 33.71	70.01 / -0.03
	SFP	0.55 / 38.52	0.17 / 36.54	88.24 / -3.07	8.90 / 39.73	0.38 / 39.31	69.62 / -0.58
	HOS	0.53 / 40.97	0.15 / 42.55	90.18 / -0.95	8.87 / 39.99	0.38 / 39.51	64.34 / -8.12
	LFB	0.54 / 40.19	0.14 / 46.12	89.99 / -1.15	9.40 / 36.21	0.04 / 93.00	60.94 / -13.04
	Evolution	0.45 / 49.87	0.14 / 48.83	91.77 / 0.80	8.11 / 45.11	0.36 / 42.54	69.03 / -1.43
	AutoMC	**0.55 / 39.17**	**0.18 / 31.61**	**92.61 / 1.73**	**8.18 / 44.67**	**0.42 / 33.23**	**70.73 / 0.99**
	RL	0.20 / 77.69	0.07 / 75.09	87.23 / -4.18	8.11 / 45.11	0.44 / 29.94	63.23 / -9.70
	Random	0.22 / 75.95	0.06 / 77.18	79.50 / -12.43	8.10 / 45.15	0.33 / 47.80	68.45 / -2.25
≈ 70	LMA	0.27 / 70.40	0.08 / 72.09	75.25 / -17.35	4.44 / 69.98	0.19 / 69.90	41.51 / -40.73
	LeGR	0.27 / 70.03	0.16 / 41.56	85.88 / -5.67	4.43 / 69.99	0.45 / 28.35	69.06 / -1.38
	NS	0.27 / 70.05	0.06 / 78.77	85.73 / -5.83	4.43 / 70.01	0.27 / 56.77	68.98 / -1.50
	SFP	0.29 / 68.07	0.09 / 67.24	86.94 / -4.51	4.47 / 69.72	0.19 / 69.22	68.15 / -2.68
	HOS	0.28 / 68.88	0.10 / 63.31	89.28 / -1.93	4.43 / 70.05	0.22 / 64.29	62.66 / -10.52
	LFB	0.27 / 70.03	0.08 / 71.96	90.35 / -0.76	6.27 / 57.44	0.03 / 95.2	57.88 / -17.35
	Evolution	0.44 / 51.47	0.10 / 63.66	89.21 / -2.01	4.14 / 72.01	0.22 / 64.30	60.47 / -13.64
	AutoMC	**0.28 / 68.43**	**0.10 / 62.44**	**92.18 / 1.25**	**4.19 / 71.67**	**0.32 / 49.31**	**70.10 / 0.11**
	RL	0.44 / 51.52	0.10 / 63.15	88.30 / -3.01	4.20 / 71.60	0.19 / 69.08	51.20 / -27.13
	Random	0.43 / 51.98	0.13 / 52.53	88.36 / -2.94	5.03 / 65.94	0.28 / 55.37	51.76 / -25.87

表 3: 使用 ResNets 在 CIFAR-10 上的压缩结果和 VGGs 在 CIFAR-100 上的压缩结果，目标剪枝率设置为 40%。表格中所有数据都被形式化为 PR(%) / FR(%) / Acc.(%)。

Algorithm	ResNet-20 on CIFAR-10	ResNet-56 on CIFAR-10	ResNet-164 on CIFAR-10	VGG-13 on CIFAR-100	VGG-16 on CIFAR-100	VGG-19 on CIFAR-100
LMA	41.74 / 42.84 / 77.61	41.74 / 42.93 / 79.61	41.74 / 42.96 / 58.21	40.07 / 40.29 / 47.16	40.11 / 40.26 / 42.11	40.12 / 40.25 / 40.02
LeGR	39.86 / 21.20 / 89.20	40.02 / 25.76 / 90.69	39.99 / 33.11 / 83.93	40.00 / 12.15 / 70.80	39.99 / 11.55 / 69.97	39.99 / 11.66 / 69.64
NS	40.05 / 44.12 / 88.78	40.02 / 55.68 / 89.19	39.98 / 51.13 / 83.84	40.01 / 31.19 / 70.48	40.00 / 33.71 / 70.01	40.00 / 41.34 / 69.34
SFP	38.30 / 35.49 / 87.81	38.52 / 36.54 / 88.24	38.58 / 36.88 / 82.06	39.68 / 39.16 / 70.69	39.73 / 39.31 / 69.62	39.76 / 39.40 / 69.42
HOS	40.12 / 39.66 / 88.81	40.97 / 42.55 / 90.18	41.16 / 43.50 / 84.12	40.06 / 39.36 / 64.13	39.99 / 39.51 / 64.34	40.01 / 39.13 / 63.37
LFB	**40.38 / 45.80 / 91.57**	40.19 / 46.12 / 89.99	40.09 / 76.76 / 24.17	37.82 / 92.92 / 63.04	36.21 / 93.00 / 60.94	35.46 / 93.05 / 56.27
Evolution	49.50 / 46.66 / 89.95	49.87 / 48.83 / 91.77	49.95 / 49.44 / 87.69	45.15 / 35.58 / 62.95	45.11 / 42.54 / 69.03	45.19 / 36.64 / 63.30
Random	75.94 / 74.44 / 78.38	75.95 / 77.18 / 79.50	75.91 / 78.08 / 59.37	45.18 / 24.04 / 62.02	45.15 / 47.80 / 68.45	45.11 / 33.06 / 68.81
RL	77.87 / 69.05 / 84.28	77.69 / 75.09 / 87.23	77.23 / 83.27 / 74.21	45.20 / 26.00 / 62.36	45.11 / 29.94 / 63.23	45.14 / 38.78 / 68.31
AutoMC	38.73 / 30.00 / 91.42	**39.17 / 31.61 / 92.61**	**39.30 / 40.76 / 88.50**	**44.60 / 34.43 / 71.77**	**44.67 / 33.23 / 70.73**	**44.68 / 35.09 / 70.56**

4 实验

在这一部分中，我们来测试 AutoMC 的性能。我们首先将 AutoMC 与人工设计的压缩方法进行比较，分析 AutoMC 的应用价值及其搜索空间设计的合理性（节 4.2）。其次，我们将 AutoMC 与经典 AutoML 算法进行比较，以测试其搜索策略的有效性（节 4.3）。然后，我们将 AutoMC 搜索到的压缩方案转移到其他神经模型以检查其可迁移性（节 4.4）。最后，我们进行消融研究，分析基于领域知识和渐进式搜索策略的嵌入式学习方法对 AutoMC 整体性能的影响（节 4.5）。

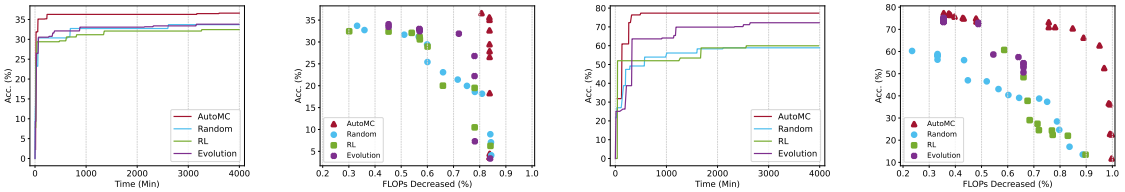
我们使用 Pytorch 实现了所有算法，并使用 RTX 3090 GPU 进行了所有实验。

4.1 实验设置

比较算法. 我们将 AutoMC 与两种流行的 AutoML 搜索策略进行比较：结合循环神经网络控制器的强化学习搜索策略 [7] 和基于遗传的多目标优化搜索策略 [7]，以及 AutoML 中常用的基准，随机搜索。为了使这些 AutoML 算法能够处理我们的自动模型压缩问题，我们将它们的搜索空间设置为 $\mathbb{S}(L=5)$。此外，我们采用了 6 种最先进的人工设计的压缩方法：LMA [28]、LeGR [6]、NS [23]、SFP [9] , HOS [2] 和 LFB [15]，作为基准，展示自动模型压缩的重要性。

压缩任务. 我们构建了两个实验来测试 AutoML 算法的性能。**Exp1:** D=CIAFR-10, M= ResNet-56, γ=0.3; **Exp2:** D= CIAFR-100, M=VGG-16, γ=0.3，其中 CIAFR-10 和 CIAFR-100 [14] 是两个常用的图像分类数据集，而 ResNet-56 和 VGG-16 是两种流行的 CNN 网络架构。

为了提高执行速度，我们在实验中从 D 中抽取 10% 的数据来执行 AutoML 算法。执行 AutoML 算法后，我们选择帕累托最优压缩方案 $PR \geq \gamma$ 进行评估。对于现有的压缩方法，我们应用网格搜索来获得它们的最佳超参数设置，并将它们的参数缩减率设置为 0.4 和 0.7 来分析它们的压缩性能。



(a) Achieved highest accuracy score (Exp1) **(b)** Final Pareto front (Exp1) **(c)** Achieved highest accuracy score (Exp2) **(d)** Final Pareto front (Exp2)

图 4: 不同 AutoML 算法在 Exp1 和 Exp2 上搜索到的帕累托最优结果。

此外，为了评估 AutoML 算法搜索的压缩方案的可迁移性，我们设计了两个迁移实验。我们将在 ResNet-56 上搜索到的压缩方案转移到 ResNet-20 和 ResNet-164，并将方案从 VGG-16 转移到 VGG-13 和 VGG-19。

实现细节. 在 AutoMC 中，嵌入大小设置为 32。$\mathcal{NN}_{exp}$ 和 $\mathcal{F}_{mo}$ 使用 Adam 进行训练，学习率为 0.001。AutoMC 搜索 3 个 GPU 天后，我们选择帕累托最优压缩方案作为最终输出。对于比较的 AutoML 算法，我们遵循他们论文中报告的实现细节，并控制每个 AutoML 算法的运行时间相同。图 6 给出了 AutoMC 搜索到的最佳压缩方案。

4.2 与现有压缩方法的比较

表 2 给出了 AutoMC 的性能和不同任务上现有的压缩方法。我们可以观察到 AutoMC 设计的压缩方案在所有任务中都超过了手动设计的方案。这些结果证明 AutoMC 具有很大的应用价值。它有能力帮助用户自动搜索更好的压缩方案来解决特定的压缩任务。

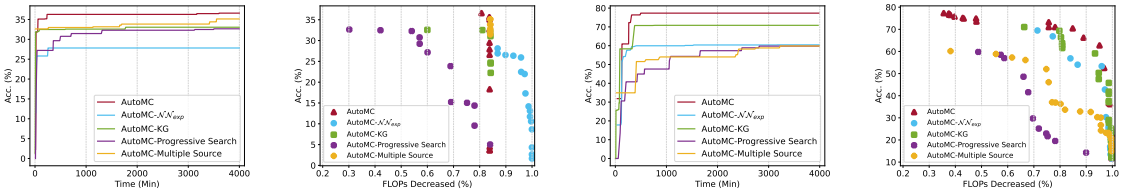
此外，实验结果告诉我们：(1) 压缩策略可能在较小的参数缩减率（PR）下表现更好。以 ResNet-56 在 CIFAR-10 上使用 LeGR 的结果为例，当 PR 为 0.4 时，参数量每下降 1%，模型性能平均下降 0.0088%；但是，当 PR 变大时，参数量每下降 1%，模型性能下降 0.0737%。(2) 不同的压缩策略可能适用于不同的压缩任务。例如，当 $PR = 0.4$ 时，LeGR 的表现优于 HOS，而当 $PR = 0.7$ 时，HOS 的表现优于 LeGR。基于以上两点，针对给定的压缩任务，结合多种压缩策略和细粒度压缩可能会取得更好的效果。这与我们设计 AutoMC 搜索空间的思路是一致的，也进一步证明了 AutoMC 搜索空间设计的合理性。

4.3 与 AutoML 算法的比较

表 2 给出了不同 AutoML 算法在不同压缩任务上的性能。图 4 提供了最佳压缩方案（准确度得分最高的帕累托最优方案）和 AutoML 算法搜索到的所有帕累托最优方案的性能。我们可以观察到，强化学习算法在很早的阶段表现良好，但在后期的性能提升却远远落后于其他 AutoML 算法。进化算法在两个实验中都优于除 AutoMC 之外的其他算法。至于 Random 算法，它的性能在整个过程中一直在上升，但仍然比大多数算法差。与现有的 AutoML 算法相比，AutoMC 可以更快地搜索到更好的模型压缩方案，更适合包含大量候选者的搜索空间。这些结果证明了 AutoMC 的有效性及其搜索策略设计的合理性。

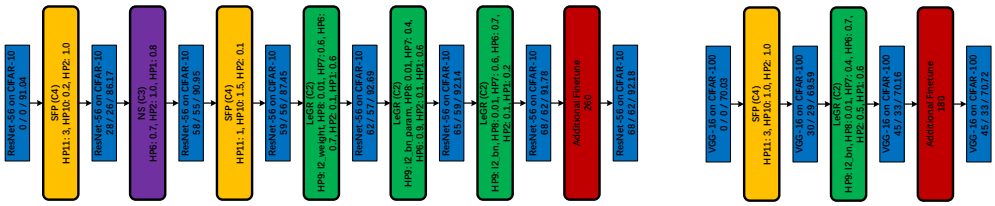
4.4 迁移研究

表 3 显示了从 ResNet-56 和 VGG-16 迁移的不同模型的性能。我们可以观察到 ResNet-20 在 CIFAR-10 上的性能 LFB 优于 AutoMC。我们认为原因是 LFB 适合于处理规模较小的模型。很明显，随着模型规模的增加，LFB 的性能逐渐下降。例如，LFB 在 CIFAR-10 上使用 ResNet-20 可以达到 91.57% 的准确率，但在 CIFAR-10 上使用 ResNet-164 只能达到 24.17% 的准确率。除此之外，AutoMC 设计的压缩方案在所有



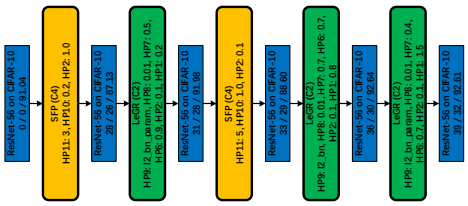
(a) Achieved highest accuracy score (Exp1) **(b)** Final Pareto front (Exp1) **(c)** Achieved highest accuracy score (Exp2) **(d)** Final Pareto front (Exp2)

图 5: 不同版本的 AutoMC 在 Exp1 和 Exp2 上的帕累托最优搜索结果。

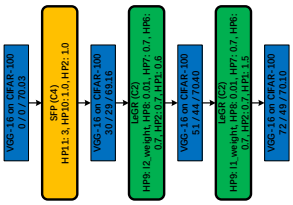


(a) Scheme on ResNet-56, PR = 40%

(c) Scheme on VGG-16, PR = 40%



(b) Scheme on ResNet-56, PR = 70%



(d) Scheme on VGG-16, PR = 70%

图 6: AutoMC 搜索的压缩方案，额外的微调将添加到序列的末尾，为比较补足微调轮数。

任务中都超过了人工设计的方案。这些结果证明 AutoMC 具有很好的可迁移性，它能够帮助用户自动搜索适用于不同尺度的模型更好的压缩方案。

此外，实验结果表明，相同的压缩策略可以在不同尺度的模型上实现不同的性能。除了上面的 LFB 和 AutoMC 示例之外，LeGR 在使用 ResNet-20 时的性能优于 HOS，而在使用 ResNet-164 时，HOS 的性能优于 LeGR。综上所述，多种压缩策略相结合，对不同尺度的模型进行细粒度压缩，可以获得更稳定、更有竞争力的性能。

4.5 消融研究

我们使用以下四种 AutoMC 变体进一步研究了基于知识的嵌入学习方法、基于经验的嵌入学习方法和渐进式搜索策略（我们算法的三个核心组件）对 AutoMC 性能的影响，从而验证了本文提出的创新。

1 *AutoMC-KG*. 此版本的 AutoMC 移除了知识图谱嵌入方法。

2 *AutoMC-*$\mathcal{NN}_{exp}$. 此版本的 AutoMC 删除了基于实验经验的嵌入方法。

3 *AutoMC-Multiple Source*. 此版本的 AutoMC 仅使用一种策略 LeGR 来构建搜索空间。

4 *AutoMC-Pregressive Search*. 此版本的 AutoMC 将渐进式搜索策略替换为结合循环神经网络控制器的强化学习搜索策略。

对应的结果如图 5，我们可以看到 AutoMC 的性能比 *AutoMC-KG* 和 *AutoMC-*$\mathcal{NN}_{exp}$ 好很多，后面二者在学习嵌入时忽略压缩策略的知识图或实验经验。这一结果向我们展示了在 AutoMC 中充分考虑两种关于压缩策略的知识，对于有效嵌入学习的重要性和必要性。我们提出的知识图嵌入方法可以探索搜索空间中压缩策略之间的差异和联系，基于实验经验的嵌入方法可以揭示压缩策略的性能特征。两种嵌入学习

方法可以相互补充，帮助 AutoMC 更好、更全面地理解搜索空间组件。

此外，我们注意到 *AutoMC-Multiple Source* 的性能比 AutoMC 差。*AutoMC-Multiple* 只使用一种压缩方式来完成压缩任务，结果表明使用多源压缩策略构建搜索空间的重要性。

我们还观察到 *AutoMC-Progressive Search* 的性能比 AutoMC 差得多。强化学习的非渐进式搜索过程，即仅搜索、评估和分析完整的压缩方案，在自动压缩方案设计问题任务中表现更差。它未能有效地利用历史评估细节来提高搜索效果，因此不如 AutoMC 有效。

5 结论

在本文中，我们提出了 AutoMC ，它可以根据用户的要求自动设计最优压缩方案。AutoMC 创新性地引入领域知识辅助搜索策略，深入了解每种压缩策略的潜在特点和优势，从而更合理、更轻松地设计压缩方案。此外，AutoMC 提出了渐进式搜索空间扩展的思想，可以选择性地探索有价值的搜索区域，并通过更细粒度的分析逐步提高搜索方案的质量。这种策略可以减少无用的评价，提高搜索效率。大量的实验结果表明，现有压缩方法的组合可以创建更强大的压缩方案，以上两项创新使得 AutoMC 比现有的 AutoML 方法更高效。

参考文献

[1] Irwan Bello et al. “Neural Optimizer Search with Reinforcement Learning”. In: *ICML*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 459–468.

[2] Christos Chatzikonstantinou et al. “Neural Network Compression Using Higher-Order Statistics and Auxiliary Reconstruction Losses”. In: *CVPR*. 2020, pp. 3077–3086.

[3] Daoyuan Chen et al. “AdaBERT: Task-Adaptive BERT Compression with Differentiable Neural Architecture Search”. In: *IJCAI*. Ed. by Christian Bessiere. ijcai.org, 2020, pp. 2463–2469.

[4] Yukang Chen et al. “RENAS: Reinforced Evolutionary Neural Architecture Search”. In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 4787–4796.

[5] Ran Cheng et al. “A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization”. In: *IEEE Trans. Evol. Comput.* 20.5 (2016), pp. 773–791.

[6] Ting-Wu Chin et al. “Towards Efficient Model Compression via Learned Global Ranking”. In: *CVPR*. 2020, pp. 1515–1525.

[7] Yang Gao et al. “Graph Neural Architecture Search”. In: *IJCAI*. Ed. by Christian Bessiere. ijcai.org, 2020, pp. 1403–1409.

[8] Ariel Gordon et al. “MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks”. In: *CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 1586–1595.

[9] Yang He et al. “Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks”. In: *IJCAI*. Ed. by Jérôme Lang. 2018, pp. 2234–2240.

[10] Yuval Heffetz et al. “DeepLine: AutoML Tool for Pipelines Generation using Deep Reinforcement Learning and Hierarchical Actions Filtering”. In: *KDD*. Ed. by Rajesh Gupta et al. ACM, 2020, pp. 2103–2113.

[11] Benoit Jacob et al. “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference”. In: *CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 2704–2713.

[12] Aaron Klein et al. “Meta-Surrogate Benchmarking for Hyperparameter Optimization”. In: *NeurIPS*. Ed. by Hanna M. Wallach et al. 2019, pp. 6267–6277.

[13] Tamara G. Kolda and Brett W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Rev.* 51.3 (2009), pp. 455–500.

[14] A. Krizhevsky and G. Hinton. “Learning multiple layers of features from tiny images”. In: *Handbook of Systemic Autoimmune Diseases* 1.4 (2009).

[15] Yawei Li et al. “Learning Filter Basis for Convolutional Neural Network Compression”. In: *ICCV*. 2019, pp. 5622–5631.

[16] Yuchao Li et al. “Exploiting Kernel Sparsity and Entropy for Interpretable CNN Compression”. In: *CVPR*. 2019, pp. 2800–2809.

[17] Shaohui Lin et al. “Towards Convolutional Neural Networks Compression via Global Error Reconstruction”. In: *IJCAI*. Ed. by Subbarao Kambhampati. IJCAI/AAAI Press, 2016, pp. 1753–1759.

[18] Shaohui Lin et al. “Towards Optimal Structured CNN Pruning via Generative Adversarial Learning”. In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 2790–2799.

[19] Shaohui Lin et al. “Towards Optimal Structured CNN Pruning via Generative Adversarial Learning”. In: *CVPR*. Computer Vision Foundation / IEEE, 2019, pp. 2790–2799.

[20] Yankai Lin et al. “Learning Entity and Relation Embeddings for Knowledge Graph Completion”. In: *AAAI*. Ed. by Blai Bonet and Sven Koenig. 2015, pp. 2181–2187.

[21] Hanxiao Liu, Karen Simonyan, and Yiming Yang. “DARTS: Differentiable Architecture Search”. In: *ICLR*. OpenReview.net, 2019.

[22] Zechun Liu et al. “MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning”. In: *ICCV*. IEEE, 2019, pp. 3295–3304.

[23] Zhuang Liu et al. “Learning Efficient Convolutional Networks through Network Slimming”. In: *ICCV*. 2017, pp. 2755–2763.

[24] Masahiro Nomura et al. “Warm Starting CMA-ES for Hyperparameter Optimization”. In: *AAAI*. AAAI Press, 2021, pp. 9188–9196.

[25] Asaf Noy et al. “ASAP: Architecture Search, Anneal and Prune”. In: *AISTATS*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 493–503.

[26] Esteban Real et al. “Regularized Evolution for Image Classifier Architecture Search”. In: *AAAI*. AAAI Press, 2019, pp. 4780–4789.

[27] M. Sanaullah. “A Review of Higher Order Statistics and Spectra in Communication Systems”. In: *Global Journal of Science Frontier Research* (May 2013), pp. 31–50. DOI: 10.34257/GJSFRAVOL13IS4PG31.

[28] Zhenhui Xu et al. “Light Multi-Segment Activation for Model Compression”. In: *AAAI*. 2020, pp. 6542–6549.

[29] Anatoly Yakovlev et al. “Oracle AutoML: A Fast and Predictive AutoML Pipeline”. In: *Proc. VLDB Endow.* 13.12 (2020), pp. 3166–3180.

[30] Aojun Zhou et al. “Incremental Network Quantization: Towards Lossless CNNs with Low-precision Weights”. In: *ICLR*. OpenReview.net, 2017.