基于图形的进化编码卷积神经网络架构设计

William Irwin-Harris 、 Yanan Sun †、 Bing Xue和 Mengjie Zhang

惠灵顿维多利亚大学工程与计算机科学学院,惠灵顿 6140,新西兰 †四川大学计算机科学学院,成都,中国,610064 @ecs.vuw.ac.nz
† ysun@scu.edu.cn

摘要 卷积神经网络 (CNN) 在一系列数据集的图像分类中展示了高效的性能。只有选择合适的架构,CNN 才能获得最佳性能,这取决于可用训练数据的数量和性质。文献中的许多最先进的架构都是由人类研究人员手工打造的,但这需要 CNN 方面的专业知识、领域知识或反复试验,通常会使用昂贵的资源。最近基于进化深度学习的工作提供了一种替代方案,其中将进化计算 (EC) 应用于自动架构搜索。进化深度学习的一个关键组成部分是选择的编码策略;然而,以前在 EC 中进行 CNN 编码的方法通常在可以表示的架构方面有限制。在这里,我们提出了一种基于有向无环图表示的编码策略,并介绍了一种使用这种编码随机生成 CNN 架构的算法。与之前的工作相比,我们提出的编码方法更通用,能够表示任意连接结构和无限深度的 CNN。

我们使用随机搜索来证明其有效性,其中评估了 200 个随机生成的 CNN 架构。为了提高计算效率,仅使用 10% 的 CIFAR-10 训练数据对 200 个 CNN 进行训练;然后在完整的训练集上重新训练三个表现最好的 CNN。结果表明,尽管随机搜索方法简单且数据集减少,但与手动设计的体系结构相比,所提出的表示和初始化方法可以实现有希望的准确性。我们打算未来的工作可以通过使用这种编码应用进化搜索来改进这些结果。

一、引言

深度学习 [1] 在图像分类 [2] 方面取得了巨大成功,卷积神经网络 (CNN) 成为模型的主要选择。用于图像识别的深度学习的进步在很大程度上是由新的 CNN 架构的发展推动的:在引入带有 LeNet5 [3] 的 CNN 之后,其他成功的架构包括 AlexNet [2]、VGG [4]、GoogleNet [5]]、ResNet [6] 和 DenseNet [7]。

然而,设计 CNN 仍然是一个具有挑战性的问题:它不仅需要熟悉 CNN 的机器学习研究人员的专业知识,还需要特定问题领域的知识才能获得最佳结果 [8]。

通常,还需要进行大量的反复试验。因此,为特定应用设计新的 CNN 通常是耗时且成本高昂的,因为需要人工专业知识。此外,即使决定使用已经开发(例如已发布)的 CNN 架构来解决现实世界的问题,这也可能不会产生最佳结果,因为该架构并不是专门为目标应用程序及其相应的应用程序设计的数据。特别是对于少量的训练数据,相对较浅的 CNN 可能会给出最好的结果。相反,如果有大量可用的训练数据,最好的结果可能是由非常深的 CNN 获得的。

由于这些与手动选择 CNN 架构相关的挑战,用于派生新架构的自动技术已成为近期工作的主题。该领域的现有工作通常采用两种关键架构搜索策略中的任何一种。其中之一是强化学习,这是神经架构搜索 [9]、MetaQNN [10] 和 Block-QNN [11] 等方法采用的方法。这些方法取得了很好的效果,但通常需要大量的计算资源。另一个关键策略是基于进化深度学习,其中采用进化计算方法来推导出新的深层结构。此类工作包括遗传 CNN 方法 [12]、大规模进化方法 [13]、层次表示方法 [14]、基于笛卡尔遗传规划的方法 [15]、CNN-GA 方法 [16]和 GP-CNAS 方法 [17]。

进化架构搜索算法在基准数据集上取得了很好的结果,包括 CIFAR-10 和 CIFAR-100 [18],证明了这种方法的潜力。

但是,仍然存在一些限制;我们认为这主要是由于编码策略的选择,这是进化深度学习的关键组成部分。遗传 CNN [12] 和 CGP-CNN 方法 [15] 都限制了进化 CNN 的最终深度,因此用户必须在开始进化过程之前指定最大深度。对于 CGP-CNN [15],这是由于使用了具有固定行数和列数 (Nr Nc) 的网格的标准笛卡尔遗传编程编码。 CGP-CNN [15] 的另一个限制是没有使用交叉算子;该算法仅使用点突变来进化架构。在遗传 CNN [12] 的情况下,由于固定长度二进制字符串的编码策略,最大 CNN 深度在算法运行之前也是固定的。

更广泛地说,这些限制可以描述为显示

978-1-7281-2153-6/19/$31.00 c 2019 IEEE

算法是半自动的而不是全自动的;也就是说,在前面两个示例中,仍然需要用户输入来指定编码的最大深度。层次表示方法 [14] 也存在类似的要求,其中用户必须预先确定层次表示中的级别数,以及图中每个级别的节点数。

尽管 CNN-GA [16] 方法是全自动的,但由于它使用了构建块链表的编码,这些构建块在演化过程中可能会增长到任意深度,因此它生成的体系结构的连接结构受到限制。

链表编码中的每个构建块都是一个池化层或一个“跳跃层”,其中一个跳跃层包括两个卷积层和一个跳跃连接[16];该设计以 ResNet [6] 为模型。这个搜索空间取得了很好的效果,但这意味着 CNN-GA [16] 不能生成任意图结构的 CNN 架构。使用不同于 ResNet [6] 的复杂图形结构的著名架构包括 DenseNet [7] 和由 Block-QNN 强化学习方法 [11] 设计的 Block-QNN-S 单元。

我们在本文中的目标是通过开发一种新的编码策略来解决这些限制,该策略可以表示任意图形结构和无限深度的 CNN 架构。本文的主要贡献如下:

1) 我们提出了一种新颖的编码策略,旨在应用于进化深度学习,其中每个 CNN 架构都表示为有向无环图。与为进化搜索设计的其他表示相比,我们对搜索空间应用的约束要少得多。我们相信这可能使搜索算法能够派生出新的结构,例如跳过连接的新应用。

2) 我们引入了一种根据提议的编码随机生成架构的方法,这样该方法既可以用作进化算法的种群初始化组件,也可以单独用于随机搜索。我们提出的初始化方法旨在对搜索空间应用最小限制,同时避免生成无效或明显次优的架构。

3) 使用简单的随机搜索证明了这种编码策略和相关初始化方法的有效性,在搜索过程中仅使用了最终训练数据的 10%。我们表明,即使在搜索空间很大、搜索过程中训练数据有限以及搜索算法简单的情况下,表示也能够达到有希望的准确性。

二。背景

在本节中,我们将简要回顾 CNN 的核心组件,尤其是跳跃连接的使用。

A. 标准 CNN 层

CNN 的两个关键组件是卷积层和池化层。由于本文侧重于图像识别,我们考虑二维卷积核的情况。为了二维情况下,每一层的输出可以描述为一个C M N数组,其中C是通道数,M和N是二维数组的宽和高。每个这样的二维数组通常被称为特征图。一层的输入可以直接取自上一层的输出,也可以通过求和、拼接等操作构造;这两个操作将在下一节中讨论。此外,层可以直接连接到图像像素,在这种情况下,输入通道的数量通常为三个(用于红色、绿色和蓝色分量)。

卷积层通过应用卷积核产生一堆输出特征图,卷积核通常是奇数维的方阵;典型尺寸为 3 3、5 5 和 7 7。通过将内核值与输入(输入特征图或图像)中的相应数组元素逐元素相乘,然后对这些乘积求和以产生内核的输出,从而应用卷积核。在网络训练期间,每个卷积层学习其卷积核的值,其中核的数量由输入数据的维度和输出特征图的数量决定。

输出特征图的数量是该层的一个参数。

池化层的应用类似于卷积层,但不是使用卷积核,而是应用运算符来聚合空间区域上的输入数据;通常这是算术平均值(平均池化)或最大值(最大池化)。大多数情况下,池化层用于空间降维;通道数不变。

CNN 可以选择在 CNN 结构的末端包括一个或多个完全连接(密集)层,紧接在输出层之前。这些层遵循多层前馈神经网络的标准结构,其中层中的每个节点都连接到前一层中的每个节点。然而,在我们提出的编码方法中,我们避免使用全连接层。这与相关工作[15]、[16]中采用的方法一致,全连接层由于其密集连接而容易过度拟合[19]。

因此,对 CNN 架构进行编码的任务可以描述为指定层数、它们的连接结构以及每一层的类型和参数。

B. 跳过连接

最近的 CNN 架构中的一项关键技术是跳过连接(也称为快捷连接)。最直接的 CNN 架构由层的线性序列组成,其中每一层都直接从前一层获取输入。然而,由于包含跳跃连接,层的输入可以指定为紧邻的前一层输出的总和以及序列中较早层的输出的总和。 skip connections 的有效性在 ResNet [6] 中的成功应用得到了凸显。在 ResNet 中,该结构基于“残差块”的重复序列(每个残差块

包含两个卷积层和一个跳跃连接)[6]。
在 [20] 中讨论了跳过连接在缓解梯度消失问题（这在深度神经网络中很常见)方面的好处,并且在 [21] 中也通过实验证明了它们在深度学习中的有效性。

跳跃连接的概念可以通过将 CNN 视为潜在任意结构的有向无环图来推广。在这种情况下,我们认为卷积层或池化层的输入可能是任何其他层的输出,或者其他层输出的求和或串联,但要求不形成循环。

求和运算符执行两个 C M N 数组的逐元素加法;即,两个多维数组必须具有相同的空间维度和通道数。串联运算符连接C1 M N 和C2 M N 数组的通道以生成大小为 (C1 + C2) M N 的输出数组。与求和一样,两个数组的空间维度必须连接时相等。

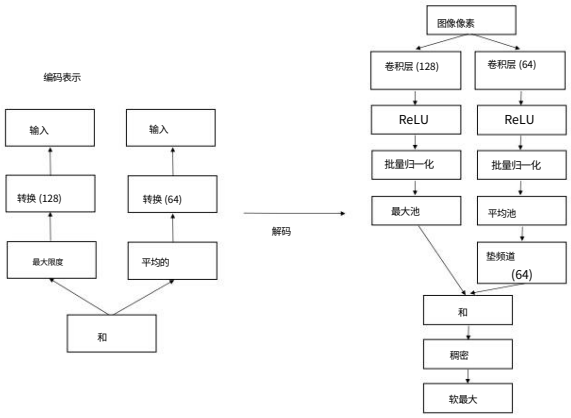

图 1. 编码表示示例及其相应的解码 CNN 架构。

三、提议的编码策略

本节讨论所提出的 CNN 编码方法,并提出一种用于随机初始化由编码描述的 CNN 架构的算法。

A. 编码策略概述如前所述,CNN 的组

件可以被视为有向无环图。在本小节中,我们详细介绍了我们提出的基于图的编码。一般图可以描述为一个有序对 G = (V,E),顶点为 V,边为 E。对于有向无环图,每条边都与一个方向相关联,并且存在一个限制,即没有从节点 v 到它自己。对于我们的 CNN 架构编码,我们注意到进一步的限制,即总是恰好有一个节点 r,这样就没有以 r 结尾的边（但可能有以 r 开始的边）。我们将其称为根节点,它代表 CNN 的输出。

在这种编码中,图中的每个节点都对应于一个构建块、一个运算符或对输入数据的引用。我们为每个节点分配一个类型,它是SUM、 CONCAT、 CONV、 MAX、 AVERAGE和INPUT 之一。此外， CONV节点还有一个 feature maps属性,指定了卷积层输出中通道的整数个数。在这种编码中,我们选择一个约定,即从节点n1到节点n2的边表示n1从 n2 获取其输入数据。这与网络中数据流的方向相反。也就是说,边缘从网络的输出层指向输入数据（如图 1 的左侧所示）。这种约定是任意的,但便于描述作用于该结构的算法。

每个节点类型都有一个固定的元数,它指定从该节点开始所需的边数。请注意,一个节点可能有任意数量的边终止于该节点。

INPUT类型的节点的元数为 0;在下文中,我们也将INPUT节点称为叶节点。我们定义SUM和CONCAT节点的元数为 2,所有其他节点的元数为 1。

任意数量的数组的求和和串联仍然可以通过链接SUM和CONCAT节点来表示。

B. 表示和解码细节

在本小节中,我们提供了有关图形结构的更多详细信息,并描述了用于将此表示（基因型)转换为最终 CNN 架构（表型)的解码方法。图 1 给出了编码表示的示例,并显示了其相应的解码 CNN 架构。

首先,我们详细介绍可用的节点类型,并证明这些设计选择的合理性。CONV节点表示最终 CNN 中的“卷积构建块”,我们将其定义为由步幅为 1 的卷积层、ReLU 激活函数 [22] 和批量归一化 [23] 组成,顺序为.这种设计的灵感来自于 CGP-CNN 方法 [15] 中使用的 ConvBlock,并且批归一化和 ReLU 组件的使用也与最近手工制作的 CNNs [6]、[7] 一致。我们使用“相同”的填充模式和 1 1 步幅,以便图像尺寸在应用卷积后不会改变。特征图的数量是可配置的,是CONV节点的一个参数。相反,我们使用固定的过滤器（内核)大小,始终为 3 3。

做出这个选择是因为 3 3 是手动设计的 CNN（例如 ResNet [6] 和 DenseNet [7])中非常常用的过滤器大小;此外,有人建议其他尺寸的过滤器可以用 3 3 个过滤器的堆栈代替,而不会失去代表性 [24]。因此,使用固定的过滤器大小可以减少搜索空间。

MAX和AVERAGE节点类型分别表示最大和平均池化层。我们对最大和平均池化层都使用 2 的内核大小和 2 步长,因此这些层总是将输入数据的空间维度减半。然后可以通过堆叠这些池化层来实现更大的维度缩减。
INPUT节点类型用于从应用 CNN 的训练或测试数据中引用当前图像（或图像的小批量）。

从概念上讲,只需要一个INPUT节点;但是,算法实现允许多个INPUT节点会很方便,所有这些节点都被认为是等效的。

SUM和CONCAT节点分别表示求和和串联运算。在解码步骤中,如果SUM或CONCAT节点的两个输入的维度不相等,则添加最大池化层以对较大的特征图进行下采样,从而使两个特征图的维度变得相等。这是基于 [15] 和 [16] 中使用的过程。

在解码SUM节点时,如果两个输入的通道数不同,我们将包含全零的填充通道添加到通道数较小的输入,以使通道数相等(见图 1)。这种方法基于 ResNet 论文 [6] 中讨论的方法,该论文还建议将 1 1 卷积作为调整通道维度的替代方法。对于这种解码方法,我们选择使用零填充通道,因为与 1 1 卷积相比,这种技术产生的参数更少。

为了从编码表示构建最终的 CNN 架构,我们将图中的每个节点转换为其对应的 CNN 层,如上所述。最后,将输出节点数等于图像类数的全连接(密集)层添加到 CNN 的末尾。然后添加 softmax 激活函数作为 CNN 的最终组件。

C. 初始化算法概述

在本小节中,我们提出了一种随机生成以所提出的编码策略为代表的 CNN 架构的算法。该方法的总体框架在算法 1 中给出。

首先,我们随机选择一个整数深度 d(第 1 行),我们将其称为此初始化算法中的目标深度。这允许在生成种群时使用各种深度的 CNN,这可能会增加为目标数据集获得具有适当深度的架构的可能性。然后,该算法使用标准的先进先出 (FIFO) 队列以广度优先顺序构造图。生成一个随机节点作为根(第 6 行),然后图形在深度上迭代“增长”。对于队列中的每个节点,该算法会创建多个等于节点数量的“子”节点(第 15-32 行)。队列中的每个项目都包含一个节点以及该节点的级别;级别等于从节点到根的最小边数。电平信息用于

避免生成超过先前确定的目标深度的节点。级别为 ` = d 1 的节点被限制为只有INPUT节点作为子节点; INPUT类型的节点是叶节点,这会阻止图形进一步增长(第 16-17 行)。否则,对于 ` < d1,算法将为父节点随机生成新节点(第 19-29 行)。集合 R(第 19 行)用于确定不允许的节点类型,以避免生成无效或次优架构(详见算法 2)。

算法1:初始化方法的框架

```
1: d 生成一个大于零的随机整数深度
2:  QA新建FIFO队列
3: SA new, empty multiset 4: level
5: nonleaf False
6:生成一个随机的非空、非INPUT节点r作为
    root, 并将 (r, 0) 对添加到 Q 7:当Q 不为
空时 do (n, `) 从 Q a 中取出一个节点和级
8:      别 a 得到 n 的元数if ` 6= level then nonleaf False
9:      level ` end如果我 0
10:
11:
12:
13:
14:
15:     while i<a do if `
16:         +1= d then
17:             将 n 的边添加到新的INPUT节点else
18:
19:             R 计算一组不允许的节点类型 c 生成一个受 R 限制的随
20:             机节点if c = NULL then
21:
22:                 SS [ {n}否则
23:
24:                 添加一条从 n 到 c 的边if
25:                 type(c) 6= INPUT then
26:                     将 (c, ` + 1) 对排队到 Q 非叶 True end
27:                     if end if end if
28:
29:
30:
31:             →   i + 1
束时    结束时32: 33:结

34:将缺失的边添加到 S 中的所有节点,根为 r
35:删除多余的池化节点,从根 r 开始
36:返回r
```

我们注意到一个简单的广度优先算法,其中为每个当前节点生成新的子节点,只能生成树。为了能够生成有向无环图,我们允许生成随机节点的过程返回 NULL (第 21-22 行)。 NULL值用于指示不是向新节点添加边,而是应添加边以引用现有节点,从而生成有向无环图。因为在算法的这个阶段,图还没有完全构建,我们通过将节点添加到多重集 S 以供以后处理来推迟这个过程(第 22 行)。因此,整个算法可以描述为首先构造一棵树,然后使用多重集 S 添加更多边以生成有向无环图。否则,如果返回一个非 NULL随机生成的节点,我们添加新边,并将新节点入队。

在算法的主要部分之后,执行进一步的处理以将“缺失”的边添加到树中以生成图形（第 34 行）;这在算法 3 中有详细说明。除此之外,还需要进行后处理来验证池化节点的数量。因为每个池化节点（ MAX或AVERAGE)将空间维度减少 1/2,所以在图像维度变为 1 1 之前存在池化节点的最大数量,此时进一步池化无效。因为我们的池化节点使用的步长和内核大小为 2,所以对于正方形图像,线性序列中池化节点的最大数量可以计算为输入图像大小的以 2 为底的对数。在我们的实现中,我们通过计算池化节点的数量解决了这个问题,然后用随机生成的卷积节点替换多余的池化节点（第 35 行）。首先替换最靠近输入数据的池化节点,因为在其他操作（卷积等)之前使用池化节点可能会由于降维而丢失重要的图像数据。

D. 初始化算法细节

在本小节中,我们将详细介绍在前面讨论的初始化算法的总体框架中使用的组件。算法 2 给出了确定受限节点类型集 R 的过程,在生成新节点以扩展图时使用。

算法2:确定不允许的节点类型
输入:节点 n 级别为 `;深度d;布尔非叶
1: R; 2:
如果从 n 到INPUT节点存在边则为真;别的
错误3:
如果b 或不是非叶则
4: RR [{INPUT, REFERENCE} 5:如果结束

6:如果` +2= d 或 type(n) 2 {SUM, CONCAT}那么
7: RR [{BINARY} 8:如果结束

9:返回R

算法 2 的输入是当前节点 n（新节点将添加到该节点)及其级别 `,以及图形的目标深度 d。布尔值 nonleaf 在算法 1 中计算,并存储是否已在当前级别添加非叶节点（即除INPUT 之外的任何类型的节点） 。

该算法首先将 R 初始化为一个空集（第 1 行）。如果已经存在从 n 到INPUT节点的边（条件 1）,或者如果在当前层没有添加非叶节点（条件 2）,则将INPUT 添加为受限节点类型（第 2-5 行）).第一个条件的理由是元数 a > 1 的节点是SUM和CONCAT ,将输入图像数据与其自身相加或连接是多余的。第二个条件用来保证有向无环图的深度能够继续增长:如果这一层还没有添加非叶子节点,添加一个INPUT （即叶)节点可能会导致图形的生成在到达目标深度之前终止。

算法 2 中的第二个主要步骤确定是否允许二元运算符（即SUM和CONCAT)的节点（第 6-8 行）。第一个条件指定仅当“父”节点的级别 ` < d 2 时才允许使用二元运算符,其中 d 是图的目标深度。添加此条件是因为如果节点的级别为 ` = d2,则新的SUM或CONCAT节点将只能从图像输入像素中获取其输入。如果发生这种情况,这将是与先前讨论的相同情况,其中将产生图像像素的冗余求和或串联。第二个条件不允许SUM和CONCAT节点链。也就是说,不允许从两个二元节点之一到另一个二元节点的边。请注意,与算法 2 中讨论的先前情况不同,此限制并不是严格要求的。相反,我们将此条件作为对搜索空间的约束。如果允许任意序列的二元运算符,这可能会导致算法生成的分支数量迅速增加,从而可能导致过于复杂和次优的CNN。我们还注意到 CNN 编码本身仍然是通用的,并且由于它被设计为与进化算法一起使用,因此可以设计遗传算子,以便允许表示二元算子的节点之间的边。

算法 3 详细说明了基于“缺失”边的多重集 S 添加新边的过程。也就是说,该算法添加 S 中指定的边,使树成为有向无环图。

算法3:添加缺失边
输入:包含每个具有缺失边的节点 p 的多重集 S;有向无环图1的根节点r： for p in S do k 0

2:
3: t NULL有效
4: 假
5: while k is less than some maximum iteration count
6: do t 图中的一个随机节点,根为 r b1
7: 如果有向图中存在从 t 到 p 的路径,则为真;否则为假b2
8: 如果从 p 到 t 有一条边则为真;别的
错误的
9: 如果type(t) 6= INPUT而不是b1而不是b2那么有
10: 效 True break
11:
12: 如果kk
13: + 1结束
14: 结束时
15: 如果有效则
16: 添加一条从 p 到 t 的边else
17:
18: 添加一条从 p 到新的INPUT节点的边end if 20:
19: end for

对于多重集中的每个节点 p,该算法在图中选择一个随机节点 n（第 6 行）。由于节点 n 可能不是一个有效的选择（在下面讨论）,该算法重复循环直到找到一个有效的节点 n,直到达到某个最大迭代次数。如果超过此最大值,则添加从 p 到INPUT节点的边。添加从节点 p 到 n 的边可能无效的原因有两个:一个是可以形成循环（第 7 行）,或者从 p 到 n 的边可能已经存在（第 8 行）。此外， INPUT节点不被视为 n 的有效选择（除非在未识别有效 n 的情况下超过最大迭代次数）;这是因为此方法的目标是在图形内部的节点之间创建额外的连接（类似于跳过连接的概念,如前所述）。

算法4:生成随机节点
输入:一组 R 的受限节点类型

```
1: U {BINARY, CONV, POOL, INPUT, REFERENCE}
2:聚氨酯
3: t P 中的一个随机节点类型4: n 一个新节点
5: if t = BINARY then
6:      q如果q < 0.5则统一生成一个介于 0 和 1 之间的数字,然后n.type SUM
7:      else
8:
9:
10:          n.type CONCAT结束 if
     11: 12: else if t = CONV then
13:     n.type CONV
then    n.feature maps = 在 {64, 128, 256} 中随机选择14: 15: else if t = POOL
16:     q 统一生成一个介于 0 和 1 之间的数字if q < 0.5 then n.type MAX else
17:     n.type AVERAGE end if 21: 22: else if t = INPUT then n.type INPUT
18:          23: 24: else if t = REFERENCE then
19:
20:
25: n =空
26:如果结束
27:返回n
```

算法 4 详细说明了我们用来随机生成节点及其参数的方法;这种方法基于 CNN-GA 论文 [16] 中使用的方法,但我们引入了集合 R,它指定了不允许的节点类型。集合 R 用于构建允许节点类型的集合 P（第 1-2 行）,从中随机选择一个节点类型,均匀分布（第 3 行）。请注意,集合 U（第 1 行)包含的值与图形表示中的可用节点类型不同。也就是说,我们不在 U 中包含MAX和AVERAGE池化节点,而是简单地使用POOL类型;同样,有一个BINARY类型,而不是SUM和CONCAT类型。这样做的原因是 P 中的每种类型都被分配了相等的概率,我们希望在卷积节点、池化节点或二元运算之间统一选择。例如,如果P 中分别包含MAX和AVERAGE池化类型,则生成池化节点的概率将高于生成卷积节点的概率。我们希望避免这种情况,因为池化层会降低维度,因此如果过度使用可能会丢失信息。同样,如果二元运算（SUM或CONCAT)的概率高于卷积节点,则可能导致 CNN 架构过于复杂。

如果随机选择的节点类型为BINARY,则生成一个0到1之间（均匀分布)的随机数q。如果小于0.5,则返回一个新的SUM节点;否则,将生成一个CONCAT节点。如果选择的节点类型是CONV,则生成一个新的CONV节点,特征图的数量是随机选择的。具体来说,特征图的数量从{64、128、256}中统一选择,这是CNN-GA方法[16]使用的值。此外,这组固定的特征图数量选项有助于减少搜索空间（从而提高搜索效率）。如果随机选择的节点类型为POOL,则以0.5的概率产生一个MAX pooling节点;否则,使用AVERAGE池化节点。

如果选择的类型是INPUT,该方法只返回一个新的INPUT节点。特殊的REFERENCE类型用于实现不生成新节点的情况,而是应该添加到图中现有节点的边。在这种情况下,算法 4 只返回NULL。然后,该值被初始化方法（算法 1)用作标志,该方法更新要处理的“缺失”边的多重集 S,如前所述。

四、实验设计

一、概述

我们在 CIFAR-10 基准数据集 [18] 上测试了所提出的编码和初始化方法。CIFAR-10 包含 10 个类别,共 60,000 张 32×32 像素图像,其中 50,000 张构成训练集,10,000 张构成测试集。在架构搜索过程中,我们只使用全部 50,000 张训练图像中的 10%,以提高搜索效率。具体来说,在架构搜索过程中,使用 4,500 张图像作为训练集,500 张图像作为验证集。全部 50,000 个训练图像的这些子集是使用分层抽样过程选择的,以在训练和验证中保持类的均匀分布

套。

为了证明所提出的编码的有效性,我们使用简单的随机搜索来测试其性能,其中使用所提出的初始化方法随机生成 200 个 CNN 架构。每个 CNN 使用 4,500 张图像进行训练,并且在每个时期之后,在验证集上测试性能。对于每个架构,我们记录了达到的最大验证准确度。在缩减数据集上训练所有架构后,我们选择精度最高的三个架构并使用

表 I与同行竞

争者相比,使用建议的编码进行随机搜索所获得的结果。

	精度（%，CIFAR-10 测试集） 93.57 92.07	参数 GPU 天数	
ResNet（深度=101)[6]	93.34 94.17 96.37	1.7 米	-
ResNet（深度=1,202)[6]	95.62 92.90 94.60	10.2 米	-
VGG [4]	94.02 95.22	20.04 米	-
密集网 [7]		27.2 米	-
等级进化 [14]		-	300
块-QNN-S [11]		6.1 米	90
遗传 CNN [12]		-	17
大规模进化 [13]		5.4 米	2,750
CGP-CNN [15]		1.68 米	27
CNN-GA [16]		2.9 米	35
使用建议的编码进行随机搜索（试验 #1,最佳准确度）	92.45	145万	1.33
使用建议的编码进行随机搜索（试验 #1,最少的参数）	90.98	0.83 万	1.33
使用建议的编码进行随机搜索（试验 #2,最佳准确度）	92.57	1.14 万	1.33
使用建议的编码进行随机搜索（试验 #2,最少的参数）	91.44	0.75 万	1.33

45,000 张训练图像和 5000 张验证图像。选择三种架构（而不是单一的最佳架构)的原因是为了提高选择一种架构的可能性,该架构可以从减少的训练集推广到完整的训练集。为每个体系结构记录提供最佳验证准确性的纪元。最后,使用没有验证集的完整训练集（50,000 张图像)对每个架构进行另一次重新训练,直至达到先前确定的最大纪元;然后在测试集上计算重新训练的 CNN 的准确性。

二、参数设置

为了确定每个 CNN 架构的性能,我们使用基于 [17] 中使用的设置的训练例程和学习率计划。也就是说,我们使用随机梯度下降 [25],动量为 0.9,初始学习率为 0.1,使用 128 的小批量训练多达 200 个时期。

在第 60、120 和 160 轮之后,学习率衰减了 0.2 倍。我们使用 softmax 交叉熵损失作为损失函数。如果在训练 CNN 时出现错误,例如程序超出可用 GPU 内存,或获得 NaN（非数字)损失值,则 CNN 架构将被简单地丢弃。

为了减少过度拟合,我们应用系数为 0.5 103 的权重衰减,这是 [17] 中使用的设置。此外,我们采用了一种基于[15]和[16]中使用的方法的数据增强技术:执行随机水平翻转,并在图像每边填充 4 个像素后选择随机 32 32 裁剪。

对于随机搜索,需要为 CNN 种群初始化算法指定一个深度范围。在此实现中,我们将最小深度设置为 6,将最大深度设置为 12。虽然这限制了随机搜索期间的最大深度,但基于进化算法的搜索可以使用交叉和变异等遗传算子将表示扩展到任意深度;这将在未来的工作中进行调查。

五.实验结果

表 I 显示了使用我们提出的编码进行随机搜索以及 10 个最先进的同行竞争对手（包括手工制作的 CNN 和自动设计生成的 CNN)在 CIFAR-10 测试集（10,000 张图像)上获得的准确度算法。随机搜索显示的所有准确度都是在使用完整的 50,000 个训练图像重新训练发现的架构后在测试集上获得的准确度。

对于每个模型,显示了分类精度、可训练参数的数量（以百万计)和确定架构所需的 GPU 天数;这是一种类似于 [16] 中使用的表格格式。对于手工制作的架构，“GPU 天数”列标记为“-”。

对于我们提出的方法,我们展示了随机搜索的两个独立试验。对于每次试验,我们显示 CNN 的结果具有最佳精度和 CNN 具有最少的参数。对于所需的 GPU 天数,架构搜索本身需要大约 10 个小时,使用两个 GTX 1080 Ti GPU;最后三个架构的重新训练需要大约 12 个小时,使用一个 GPU,这样整个过程大约需要 1.33 个 GPU 天。

因此,尽管通过随机搜索实现的分类精度通常比手工制作的 CNN 和其他自动设计方法差,但这种方法需要的计算资源要少得多,并且尽管在架构搜索期间减少了数据集,但仍提供了有前途的性能。此外,该方法能够生成具有不同复杂性（参数数量)的模型;例如,一个生成的 CNN 在使用 50,000 张图像进行重新训练后,仅用 750k 个参数就达到了 91.44% 的准确率。

图 2 显示了来自随机搜索试验 #2 的最佳架构的图形表示。在图中，“MP”和“AP”分别表示最大和平均池化节点； “CB”节点代表卷积构建块,其中括号中的值是特征图的数量。

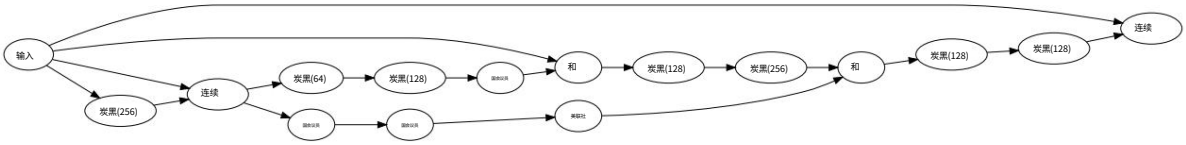

图 2. 随机搜索（试验 #2)生成的最佳 CNN 架构,在对全部 50,000 张训练图像进行重新训练后,准确率为 92.57%。

六。结论

在本文中,我们提出了一种基于有向无环图表示的 CNN 架构编码策略,旨在用于进化深度学习。

与文献中许多基于进化计算的架构搜索方法所采用的编码方法相比,我们提出的表示施加了更少的限制,从而允许表示更多的架构。

我们相信这增加了自动确定非常适合目标数据集的架构的潜力。此外,我们引入了一种随机初始化这种编码的方法,可用于进化算法中的种群初始化步骤。我们通过在架构搜索期间仅使用 10% 的 CIFAR-10 训练数据进行随机搜索来展示这种方法的潜力,但在对全部 50,000 个训练图像重新训练三个最佳发现架构后,该方法仍然取得了可喜的结果。特别是,该方法显示出生成具有各种复杂性的模型的良好能力。由于随机搜索期间使用的训练集减少,与其他方法相比,该方法也需要更少的计算时间。尽管有限制（即,简单的搜索算法和搜索期间的小数据集）,鉴于有希望的结果,如果与更复杂的架构搜索方法一起使用,所提出的 CNN 编码似乎具有获得改进性能的良好潜力。在未来的工作中,我们打算实现基于这种编码的进化深度学习方法,以进一步提高准确性。

参考

[1] Y. LeCun、Y. Bengio 和 GE Hinton，“深度学习”，《自然》，卷。 521,没有。 7553,第 436–444 页,2015 年。

[2] A. Krizhevsky、I. Sutskever 和 GE Hinton，“Imagenet classification with deep convolutional neural networks”,第 25 届神经信息处理系统国际会议论文集,第 1097-1105 页,2012 年。

[3] Y. LeCun、L. Bottou、Y. Bengio 和 P. Haffner，“基于梯度的学习应用于文档识别”,IEEE 会刊,卷。 86,第 2278–2324 页,1998 年 11 月。

[4] K. Simonyan 和 A. Zisserman，“Very Deep Convolutional Networks for Large-Scale Image Recognition”,第 32 届国际机器学习会议论文集，（法国里尔）,2015 年。

[5] C. Szegedy、W. Liu、Y. Jia、P. Sermanet、S. Reed、D. Anguelov、D. Erhan、V. Vanhoucke 和 A. Rabinovich，“深入卷积”,2015 年 IEEE计算机视觉和模式识别 (CVPR) 会议,第 1-9 页,2015 年 6 月。

[6] K. He、X. Zhang、S. Ren 和 J. Sun，“用于图像识别的深度残差学习”,2016 年 IEEE 计算机视觉和模式识别会议 (CVPR),第 770-778 页,6 月2016.

[7] G. Huang、Z. Liu、L. vd Maaten 和 KQ Weinberger，“密集连接的卷积网络”,2017 年 IEEE 计算机视觉和模式识别会议 (CVPR),第 2261-2269 页,2017 年 7 月.

[8] Y. Sun、B. Xue 和 M. Zhang，“用于图像分类的进化深度卷积神经网络”,ArXiv e-prints，p. arXiv:1710.10741,2017 年 10 月。

[9] B. Zoph 和 QV Le，“使用强化学习进行神经架构搜索”,2017 年国际学习表征会议论文集，（法国土伦）,2016 年。

[10] B. Baker、O. Gupta、N. Naik 和 R. Raskar，“使用强化学习设计神经网络架构”,2017 年国际学习表征会议论文集，（法国土伦）,2016 年。

[11] Z. Zhong, J. Yan, and C.-L. Liu，“使用 q-learning 进行实用网络模块设计”,载于 2018 年 AAAI 人工智能会议论文集（美国路易斯安那州）,2018 年。

[12] L. Xie 和 AL Yuille，“Genetic CNN”,2017 年 IEEE 计算机视觉国际会议论文集，（意大利威尼斯）,第 1388-1397 页,2017 年。

[13] E. Real、S. Moore、A. Selle、S. Saxena、YL Suematsu、J. Tan、Q. Le 和 A. Kurakin，“图像分类器的大规模演化”,机器学习论文集研究，（澳大利亚悉尼）,第 2902–2911 页，2017 年。

[14] H. Liu、K. Simonyan、O. Vinyals、C. Fernando 和 K. Kavukcuoglu，“Hierarchical representations for efficient architecture search”,2018 年机器学习研究论文集，（瑞典斯德哥尔摩）,2018 年。

[15] M. Suganuma、S. Shirakawa 和 T. Nagao，“设计卷积神经网络架构的遗传编程方法”，遗传和进化计算会议论文集,GECCO 17，（德国柏林）,pp . 497–504,ACM,2017 年。

[16] Y. Sun、B. Xue、M. Zhang 和 GG Yen，“使用遗传算法自动设计 CNN 架构进行图像分类”，
ArXiv 电子版,p. arXiv:1808.03818,2018 年 8 月。

[17] Y. Zhu、Y. Yao、Z. Wu、Y. Chen、G. Li、H. Hu 和 Y. Xu，“GP CNAS:使用遗传编程进行卷积神经网络架构搜索”,arXiv 电子版,页。 arXiv:1812.07611,2018 年 11 月。

[18] A. Krizhevsky 和 GE Hinton，“从微小图像中学习多层特征”,2009 年。在线:https: // www.cs.toronto.edu/ kriz/cifar.html。

[19] N. Srivastava、G. Hinton、A. Krizhevsky、I. Sutskever 和 R. Salakhut dinov，“Dropout:一种防止神经网络过度拟合的简单方法”,机器学习研究杂志,卷。 15,没有。 1, pp. 1929–1958, 2014。

[20] S. Hochreiter 和 J. Schmidhuber，“长短期记忆”,神经计算,卷。 9,第 1735–1780 页,1997 年 11 月。

[21] RK Srivastava、K. Greff 和 J. Schmidhuber，“训练非常深的网络”,神经信息处理系统进展，（加拿大蒙特利尔）,2015 年。

[22] X. Glorot、A. Bordes 和 Y. Bengio，“深度稀疏整流器神经网络”,第 14 届国际人工智能和统计会议论文集,2011 年。

[23] S. Ioffe 和 C. Szegedy，“批量归一化:通过减少内部协变量偏移加速深度网络训练”,第 32 届国际机器学习会议论文集,ICML 15,第 448-456 页,2015 年。

[24] C. Szegedy、V. Vanhoucke、S. Ioffe、J. Shlens 和 Z. Wojna，“重新思考计算机视觉的初始架构”,CVPR 会议记录,第 2818–2826 页,2016 年。

[25] L. Bottou，“随机梯度下降技巧”,神经网络:交易技巧:第二版,第 421-436 页,柏林,海德堡:Springer Berlin Heidelberg,2012 年。