具有基因表达的进化 NAS 蜂窝编码编程

克利福德·布罗尼-贝迪亚科
日本东京八王子市创价大学科学与工程研究生院
e18d5252@soka-u.jp

村田由纪
日本东京八王子市创价大学科学与工程研究生院
e19d5202@soka-u.jp

Luiz HB Mormille科学与工
程研究生院,日本东京八王子市 e18d5251@soka-u.jp

Masayasu Atsumi科学
与工程研究生院,日本东京八王子市
matsumi@soka.ac.jp

摘要 神经架构搜索 (NAS) 的复兴见证了经典方法,如遗传算法 (GA) 和遗传编程 (GP) 被用于卷积神经网络 (CNN) 架构。虽然最近的工作在视觉感知任务上取得了令人鼓舞的性能,但 GA 和 GP 的直接编码方案都存在功能复杂性缺陷,并且不能很好地扩展到像 CNN 这样的大型架构上。为了解决这个问题,我们提出了一种新的生成编码方案 符号线性生成编码(SLGE) 简单但功能强大的方案,它将局部图变换嵌入线性固定长度字符串的染色体中,以通过基因表达编程的进化过程。在实验中,SLGE 的有效性体现在发现可提高最先进的手工 CNN 架构在 CIFAR-10 和 CIFAR-100 图像分类任务上的性能的架构;并使用更少的 GPU 资源与现有的 NAS 方法实现具有竞争力的分类错误率。

索引词 神经结构搜索;卷积神经网络;基因表达编程;细胞编码

一、引言

从历史上看,进化神经架构搜索 (NAS) 研究在 AI 社区中引起了 30 年的兴趣 [1]。最近对深度学习自动化的兴趣日益浓厚,在过去三年中,基于强化学习 (RL) 和其他方法的新 NAS 方法取得了惊人的发展 [2]。然而,经典的 NAS 方法,如遗传算法 (GA) [3]、[4] 和遗传编程 (GP) [5] 也被用于开发用于视觉感知任务的卷积神经网络 (CNN)。除了取得有希望的结果外,经典 NAS 方法还比基于 RL 的竞争对手 [6] 消耗更少的计算资源 [4]。但是,GA 和 GP 的直接编码方案有两个固有的局限性:(1) 固定长度和 (2) 基因型和表型空间没有明显分开,这限制了它们的功能复杂性 [7]。

例如,遗传 CNN [3] 中采用的固定长度方案的染色体在大型架构上不能很好地扩展。

为了解决这些问题,EvoCNN [8] 在 GA 中引入了可变长度方案,并被 CA-CNN [4] 采用。可变长度方案用于编码类似于 GP 中非线性结构的可变形状和大小的 CNN 架构。尽管可变长度方案表现出一定程度的功能复杂性,但使用交叉操作重现并非没有困难。 EvoCNN 和 CA-CNN 及其类似的 CGP-CNN [5] 交叉限制的原因是因为它们的基因型和表型空间没有明确分开,因此,遗传修饰直接受到表型结构的约束。

DENSER [9] 将 GA 与语法进化 (GE) 相结合,以类似于自然的方式分离基因型和表型空间。但是,GE 缺乏模块化,用模块修改语法不灵活[10]。为此,我们认为分离基因型和表型空间以生成用于视觉感知任务的 CNN 架构的进化 NAS 方法的开发仍处于起步阶段。

在这项工作中,我们引入了一种新的生成编码方案,符号线性生成编码,它将细胞编码 [11] 的局部图变换嵌入到基因表达编程 [7] 的简单线性固定长度染色体中,以开发各种形状的 CNN 架构和尺寸。此外,为了使进化过程能够发现架构的新主题,常规卷积操作被用作基本搜索单元,而不是 CA-CNN [4] 中的 ResNet 和 DenseNet 块等复杂的构建块。

在实验中,初步结果通过发现 CNN 架构在 CIFAR-10 图像数据集平台上获得 3.74% 的错误率证明了所提出方法的有效性

1我们定义一个正则卷积运算为标准卷积
使用批量归一化和 ReLU 进行操作。

这项工作得到了日本 Soka Makiguchi 教育基金会的部分支持。

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

标记和转移到 CIFAR-100 时的 22.95% 错误。结果与当前自动生成的 CNN 架构具有竞争力,并且改进了最先进的手工制作的性能。本文的其余部分组织如下。第 2 节讨论相关工作,第 3 节介绍所提出方法的细节。实验结果在第 4 节中报告,第 5 节总结了本研究的未来方向。

二。相关工作

A. 进化的NAS

演化 NAS 的大多数早期工作都在小范围内演化网络架构及其连接权重 [1]。最近的神经网络,如 CNN,已经扩展了数百万个连接权重,以提高给定任务的性能。并且通过反向传播方法学习这些大型网络的连接权重优于进化方法。因此,最近的进化 NAS 工作 [2] 专注于仅进化网络架构并使用反向传播方法来优化连接权重。

通常,固定长度的染色体用于表示网络架构 [3]、[12]、[13]。但是,由于给定数据的最佳架构形状和大小未知,使用直接编码方案的可变长度染色体已被用于架构,以适应给定任务的形状和大小 [4],[14], [15]。CGP-CNN [5] 使用笛卡尔 GP 来表示可变长度结构的 CNN 架构。为了摆脱直接编码的功能复杂性缺陷,DENSER [9] 将 GA 与 GE 相结合,采用基因型和表型空间区分,并明确使用语法来生成 CNN 架构的表型。

通常,架构搜索空间分为两类:定义整个架构(宏架构)[4]、[14] 的全局搜索空间,以及用于发现可重复堆叠的微架构(单元)的基于单元的搜索空间构建整个架构 [13]、[15]。单元是一个有向无环图(DAG),它被用作构建块来形成架构。通过基于单元的方法发现的 CNN 架构是灵活的并且可以转移到其他任务 [16],并且它们比全局的 [17] 表现更好。流行的基于单元格的方法是 NASNet [16],它涉及两种类型的单元格:普通单元格和缩减单元格(用于降低特征分辨率)。

钟等。 [18] 和刘等人。 [15] 提出了类似的基于单元格的方法,但分别使用最大池和可分离卷积层来降低特征分辨率。

两个搜索空间中的基本搜索单元大多是复杂的卷积,例如 AmoebaNet [13] 中的深度可分离和非对称卷积,或 CA-CNN [4] 中的 ResNet 和 DenseNet 等复杂块。这些搜索单元降低了搜索空间的复杂性,但是,它们可能会损害灵活性并限制发现可以提高当前手工制作的。因此,在这项工作中,我们使用常规卷积(第 III-A 节)。

B.基因表达编程基因表达编程(GEP)是基

于GA和GP的成熟的基因型-表型进化方法。染色体由类似于 GA 中使用的线性固定长度基因组成,并在表型空间中开发为不同形状和大小的表达树,类似于 GP 中的解析树。这些基因在结构上以称为 Karva 符号的头部和尾部格式组织 [7]。具有不同功能的基因型和表型空间的分离使 GEP 能够以超过 GA 和 GP 的高效性执行 [19]。

Ferreira [7] 提出 GEP 中的染色体可以完全编码 ANN 以通过进化过程发现架构。到目前为止,GEP 尚未被用于像 CNN 这样的复杂架构。GEP 的致命弱点是它的 Karva 表示法,它不允许候选解决方案的分层组合,这意味着进化好的图案在后代中大多被遗传修饰破坏 [20]。因此,为了采用 GEP 进行 CNN 体系结构搜索,我们提出了一种新的生成方案,该方案将基序自然地嵌入到 GEP 的 Karva 表达中作为单个染色体,从而将新的基因型表型映射封装在进化过程的 GEP 约定中。

C. 蜂窝编码

蜂窝编码 (CE) 是一种基于简单局部图变换的生成编码,它控制节点的划分,这些节点演变成不同形状和大小的 ANN。

图形转换由具有唯一名称的程序符号表示;这些符号描绘了一个语法树(程序),它封装了 ANN 通过从具有输入和输出节点的单个初始单元 2 进化过程的开发过程。 CE 已经在广泛的问题上展示了它的效率,例如用于控制手推车上的两个杆的进化 ANN 和 6 足机器人的运动,可以在 Gruau [21] 中找到评论。

在这项工作中,我们采用了 CE 的四个转换函数并将它们嵌入到 GEP 的固定长度的单个染色体中,因为它们的转换 特别是 CPI 和 CPO [22] 生成类似于深度神经网络中突出的 ResNet 块的图案。我们简要解释了我们采用的四个功能,并参考 Gruau [11] 和 Gruau 和 Quatramaran [22] 以获得更深入的细节。图 1 描绘了 SEQ、CPO 和 CPI 转换函数的图形表示。

- 顺序划分(SEQ):将当前节点一分为二,并串联起来;子节点继承父节点的输出。
- CoPy Input division (CPI):它执行 SEQ,然后与父节点和子节点共享相同的输入。

2A unit在CE最初的想法中是单个神经元,但是在这个工作中,它代表了一个卷积层的神经元

– CoPy Output division (CPO):它执行 SEQ,然后与父节点和子节点共享相同的输出。
– 结束程序（END）:它停止开发过程。

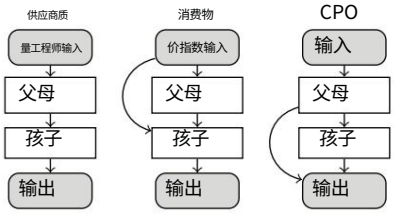


图 1:SEQ、CPI 和 CPO 转换函数示意图。

三、提议的方法

我们将 CNN 架构搜索问题表述如下。给定问题空间 Ψ = {A, S,P, tD, vD},其中 A 是架构搜索空间,S 表示搜索策略,P 表示性能度量,tD 和 vD 分别是训练和验证数据集,目标是通过搜索策略 S 找到一个更小的 CNN 架构a ∈ A,然后在数据集 tD 上对其进行训练后,它在验证数据集 vD 上最大化一些性能 P（在本例中为分类精度Pacc）。这里的较小架构意味着参数 θ 数量较少的模型。在数学上,目标函数 F 可以表示为:

$$F(\psi) = \max_{\theta,a} \; Pacc(L(a(\theta), tD|S, a \in A), vD) \quad (1)$$
$$\text{st } ModelSize(a(\theta)) \leqslant Tparams$$

其中L表示用损失函数训练模型参数θ， Tparams表示目标参数个数。在本节中,我们描述了我们提出的架构搜索空间和搜索策略。

A. 搜索空间

搜索空间中的基本搜索单元由带有批量规范的常规卷积以及 ReLU 和 CE 程序符号组成。常规卷积单元可能使进化过程能够找到新的主题来形成 CNN 架构,而不是预定义的架构（深度可分离、非对称卷积、ResNet 和 DenseNet）。由于预定义的单元降低了架构搜索空间的复杂性,它们可能会损害发现可以改进当前手工制作的新图案的灵活性。因此,我们采用基于单元格的搜索方法 [16],其中已发现单元格中的每个节点都与搜索空间中的规则卷积相关联,而边表示潜在信息流方向。继 Zhong 等人之后。 [18],我们使用最大池来降低特征图分辨率,并在必要时应用 1×1 卷积对输入深度进行下采样。

搜索空间中包含的常规卷积有:1×1、1×3、3×1 和 3×3。每个操作都有一个步幅,并应用适当的 pad 来保持空间特征图的分辨率。 CE 程序符号为:SEQ、CPI、CPO 和END。搜索空间的复杂度可以表示为：(#porgram symbols)h × (#convolution operations)h+1 × n possible architectures,其中 h 是染色体中形成基因头部的 CE 程序符号的数量n 是染色体中的基因数。例如,一条 h=2 和 n=3 的染色体,搜索空间包含 3072 种可能的结构。

B.符号线性生成编码

使用特定编码方案将网络架构编码为基因型是进化 NAS 任务的第一阶段。我们提出了一种新的生成编码方案,符号线性生成编码 (SLGE),它将 CE 的局部图变换嵌入到 GEP 的简单线性固定长度染色体中,以开发不同形状和大小的 CNN 架构。 SLGE 中的染色体可以通过 GEP 的进化过程进化出不同的基序来构建 CNN 架构。 SLGE 明确地将基因型和表型空间与自然界类似地分开,以受益于进化过程的所有优势而没有功能复杂性缺陷。值得强调的是 SLGE 的实现是多么简单,因为它是线性定长结构。我们在 geppy3 之上实现了 SLGE,geppy3 是一个基于 DEAP4框架的 GEP 库。

1)表示:遗传表示定义了表型到基因型的编码。简单但有效且高效的表示会显着影响进化过程的整体性能。在 SLGE 中,染色体的结构与 GEP 中的相似。染色体由等长定长串的基因组成。每个基因由一个由 CE 程序符号组成的头部和一个规则卷积的尾部组成。给定基因头的长度h,基因尾的长度t是h的函数,表示为:t = h + 1,因此,基因的长度为2h + 1。图2是一个例子两个基因的典型 SLGE 染色体及其表型（细胞)如图 3 所示。

2) 映射和适应度函数:映射算法将基因型（染色体)转化为表型（细胞） 每一个都被重复堆叠以构建候选 CNN 架构 然后使用适应度函数对每个架构进行几个 epoch 的训练并评估为确定其在表型空间中的适应性质量,然后将其映射回基因型空间,在那里发生遗传变异以产生下一代的后代。该表示可以在数学上表示如下:

$$Fg(\phi g):\Phi g \rightarrow \Phi p \quad (2)$$
$$Fp(\phi p):\Phi p \rightarrow R \quad (3)$$

其中Fg是将一组基因型 g ⊂ Φg 映射到表型空间Φp以形成个体架构的映射算法， Fp表示训练每个个体架构（表型)的适应度函数 p ∈ Φp

3https://geppy.readthedocs.io/en/latest/index.html
4https://deap.readthedocs.io/en/master/

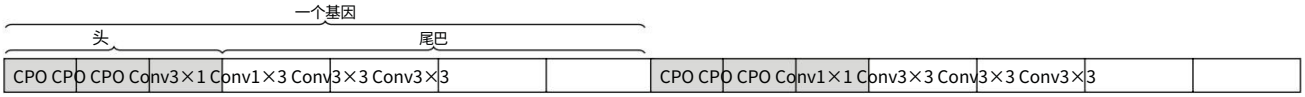


图 2:典型 SLGE 染色体的结构表示。染色体有两个等长的基因,每个基因的头部是三个CE程序符号,尾部是四个正则卷积操作。该染色体是用于构建表 II 中最佳进化发现网络的细胞基因型（图 3）。

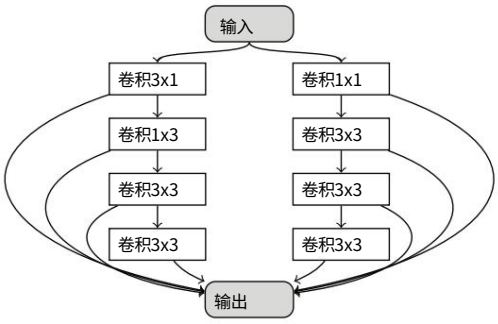


图3:图2中染色体的代表细胞,是表二中用来构建顶层网络的细胞。 （一般来说,单元格中没有后继的卷积节点被深度连接以提供输出,如果一个节点有多个前驱,则将前驱参数加在一起。这是基于单元的搜索中的常用方法 [16] .)

几个时期以确定其在适应度空间 R 中的适应度值。算法 1 和 2 是映射算法 Fg ,用于将染色体转换为 SLGE 中的细胞。这

算法以线性固定长度字符串的染色体作为输入,并产生代表候选细胞的 DAG。例如,给定图 2 中的染色体作为输入,算法 1 和 2 将产生图 3 中的细胞作为输出。

适应度函数Fp是等式 (1) 中目标函数 F 的替代项,其目的是找到一个个体架构,在模型大小约束下最大化验证数据集的分类精度。然而,由于我们在进化过程中在架构搜索空间中搜索单元,因此在其适应性评估中未考虑个体的模型大小约束。

相反,我们只对整个架构应用模型大小约束。因此,该函数只是方程 (1)中的目标函数 F,没有大小限制。损失函数用于通过在训练集和验证集上的反向传播来训练每个个体,适应度函数确定其适应度值,该值代表个体生存的概率。适应度值越高,个体就越有可能产生后代并存活到下一代。

3) 进化过程:我们采用 GEP 中的进化过程,并参考 Ferreira [7] 了解此处介绍的步骤的详细信息。

步骤 1:初始化　随机生成均匀分布的 SLGE 染色体种群。

第 2 步:映射　应用映射函数Fg将单个染色体翻译成细胞。

算法 1基因型-表型映射函数Fg输入:线性定长字符串　g ∈ Φg的染色体

输出:有向无环图 DAG (cell) 1: n ← len(　g)

//染色体中的基因数　g 2: DAG.init(null) //
用空节点初始化单元 //有输入和输出节点3: for i ← 1 to n do ←　g[i]

4:　Φ(i)　//获取染色体中的基因　g 5:创建基因
(i)中所有卷积的队列 Q 6: pnode ← Q.dequeue(0) //父节点7:
cnode ← Q.next()
//子节点8: G
← subgraph(pnode) //初始化 DAG 的子图 //for　(i) with pnode //
有输入和输出节点

9: pos ← 0
10: while |Q| > 0 do
11:　ps ← G(i)[pos]　//获取CE程序符号
12:　if ps = “END”
13:　then DAG.merge(G) //合并子图G到DAG
//在输入和输出节点
14:　返回
15:如果结束
16: TRANSFORM(G, ps, pnode, cnode) //算法2 17: pnode
← Q.dequeue(0) 18: cnode ← Q.next() pos ← pos + 1 20:
end while 21: DAG.merge(G) 22:结束23:返回DAG
19:

第 3 步:适应度　用每个单元构建候选 CNN 架构,并通过反向传播训练每个,并使用适应度函数Fp 评估其适应度。

第四步:选择　通过精英主义的轮盘赌策略选择个体形成下一代种群。

第 5 步:突变　随机突变染色体中的所有元素。必须保留结构规则（例如,卷积元素不能分配给基因头）。

第 6 步:倒置　随机倒置单个染色体基因头中的某些元素序列。

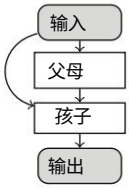
第 7 步:转置　将一些元素序列随机替换为相同色度中的连续元素

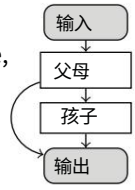
一些。必须保留结构规则。
Step 8: Recombination　两点交叉两个染色体的基因元件。由于 SLGE 染色体是

算法 2 CE 局部图变换过程。
它通过CE程序ps将基因子图G与父节点pnode进行转换生成子节点cnode。

```
1: procedure TRANSFORM(G, ps, pnode, cnode)
2:     if ps = “SEQ” then succ ←
3:         list(G.successors(pnode)) for node in succ
4:         do G.addEdge(cnode, node)
5:
6:             G.removeEdge(pnode, node)
7:结束G.addEdge(pnode, cnode) 9:结
8:         束如果

10:    if ps = “CPI” then
11:        pred = list(G.predecessors(pnode))
12:        succ = list(G.successors(pnode)) for
13:        node in pred do G.addEdge(node,
14:            cnode) 15: end for for node in succ
做G.addEdge（cnode,节点）
16:
17:
18:            G.removeEdge(pnode, node)
19: G.addEdge(pnode, cnode)结束
20:
21:如果结束
22:    如果ps = “CPO”
23:        then succ = list(G.successors(pnode))
24:        for node in succ do G.addEdge(cnode,
25:            node)
26:结束
27:        G.addEdge(pnode, cnode)
28:结束如果
29:结束过程
```





类似于 GEP 中的固定长度字符串,执行两点交叉总是会产生有效的染色体。

Step 9:如果不满足max generation则转Step 2;否则返回具有最高适应度的个体作为最佳发现细胞。

四、实验

初步实验旨在验证所提出方法的有效性,以发现在图像分类任务上表现良好的 CNN 架构的单元。我们进行了八次实验,四种不同的染色体配置各有两种（表 II）,随机搜索作为基线。搜索是在 CIFAR-10 [23] 数据集上进行的,发现的最佳架构被转移到 CIFAR-100 [23]。实验在一台 11GB GPU GeForce GTX 1080 Ti 机器上进行了 20 天。我们使用 PyTorch5 实现了所有架构,并使用 fastai6库进行了训练。这些代码可在 https://github.com/cliffbb/geppy nn 获得。

5https://pytorch.org/
6https://docs.fast.ai/

范围	价值
精英人数	
突变率	1 0.044
反转和转置率	0.1 2
反转和换位元素长度	
两点/基因重组率	0.6/0.1

表 I:搜索过程的进化参数。

染色体	网络	错误	参数
基因=2,头=2	B=[3, 3, 2]	3.93	3.2M
	B=[3, 3, 2]	3.84	2.7M
基因=2,头部=3	B=[3, 3, 1]	3.74	2.8M
	B=[2, 3, 1]	3.85	2.7M
基因=3,头部=2	B=[3, 3, 1]	4.24	3.5M
	B=[3, 3, 1]	4.03	3.4M
基因=3,头部=3	B=[3, 3, 2]	4.19	3.4M
	B=[3, 3, 2]	5.07	3.0M

表 II:四种不同染色体配置的实验结果。我们在每个配置上运行进化搜索两次,搜索后,用 C=40 从头开始训练并报告 CIFAR-10 上的分类错误率 (%)。黑体字是最好的结果。

A. 进化环境

我们使用了 20 个个体的小群体,因为 GEP 能够解决相对复杂的小群体问题 [24],并且生成 20 个个体以降低搜索过程的计算成本。用于每次运行的其他进化参数设置总结在表 I 中,这些是 GEP 中的默认设置。

B. 训练细节

每个网络以输出通道大小为 C 的 3×3 卷积干开始,然后是表示为 B=[b1, b2, b3] 的三个块　每个块 i 由bi次的重复单元组成,最大池化层插入它们之间以下采样　然后是分类器。最大池化将每个块的特征映射减少了一半。每次最大池化操作后,通道都会加倍。

在搜索过程中,CIFAR-10 50k 训练集被分成 40k 训练和 10k 验证子集并进行归一化。
我们使用了 C=16 和 B=[1, 1, 1] 的相对较小的网络。
每个候选网络都在训练子集上进行训练,并在验证子集上进行评估以确定其适应度值

在 Smith [32] 中使用 1-cycle 策略和 Adam 优化器。学习率设置为从 0.004 线性上升到 0.1,而动量在第一阶段从 0.95 线性上升到 0.85,然后在第二阶段,学习率遵循余弦退火从 0.1 到 0,而动量从 0.85 上升到 0.95同样退火。权重衰减设置为 0.0004,批量大小设置为 128,训练周期设置为 25。

搜索后,我们设置 C=40,并将每个进化发现的单元重复堆叠以构建受模型大小约束Tparams≤3.5M 的大型 CNN 架构。

我们在 800 个时期的训练和验证子集上从头开始训练每个 CNN 架构,并报告 CIFAR-10 10k 测试数据集上的分类错误。学习率保持在前 350 个时期的搜索期间,并且

模型		CIFAR-10	CIFAR-100	参数	GPU-days	方法
ResNet-110 [25]		6.61	–	1.7M	–	
ResNet-1001（激活前)[26]		4.62	22.71	10.2M	–	
分形网 [27]		5.22	23.30	38.6M	–	手工制作
密集网（k=12)[28]		4.10	20.20	7.0M	–	
DenseNet-BC (k=40) [28]		3.46	17.18	25.6M	–	
MetaQNN（顶级模型)[29]		6.92	27.14	11.2M	100	
NASv2 [6]		6.01	–	2.5M	22,400	
NASv3 [6]		4.47	–	7.1M	22,400	加强
Block-QNN-S（更多过滤器)[18]		3.54	18.06	39.8M	96	
NASNet-A (6 @ 768) [16]		3.41	–	3.3M	2,000	
ENAS + 镂空 [17]		2.89	–	4.6M	0.5	
DARTS（一阶)+镂空【30】		3.00 ± 0.14	–	3.3M	1.5	
DARTS（二阶)+镂空【30】		2.76 ± 0.09	–	3.3M	4	基于梯度
P-DARTS + 镂空 [31]		2.50	–	3.4M	0.3	
P-DARTS + 镂空 [31]		–	17.20	3.4M	0.3	
遗传 CNN [3]		7.10	29.05	–	17	
大规模进化 [14]		5.40	–	5.4M	2,750	
大规模进化 [14]		–	23.0	40.4M	2,750	
等级进化 [15]		3.63	–	–	300	进化的
CA-CNN [4]		4.30	–	2.0M	27	
CA-CNN [4]		–	20.85	5.4M	36	
变形虫网-A [13]		3.34	–	3.2M	3,150	
SLGE Networks（我们的方法）	C=40, B=[3, 3, 1]	3.74	22.95	2.8M	20	进化的
	C=40, B=[3, 3, 2]	3.84	24.47	2.7M	20	进化的
	C=40, B=[4, 4, 3]	4.47	–	3.2M	4	随机搜索

表 III:基于 CIFAR-10 和CIFAR-100 数据集。

在第一阶段从 0.00012 重置为 0.003,在 1 周期策略的第二阶段从 0.003 重置为 0。我们像 He 等人一样增加了训练子集。 [25],其他超参数与搜索期间保持不变。

C. 在 CIFAR-10 上搜索

为了研究 SLGE 的有效性,我们对四种不同配置的染色体进行了进化搜索 2 个头部为 2 和 3 的基因,以及 3 个头部为 2 和 3 的基因。每个基因在 CIFAR-10 数据集上运行两次。

具有 3 个头部的 2 个基因的染色体发现了获得 3.74% 分类错误的最佳结构（表 II）。
结果与表 III 中的其他搜索方法进行了比较。 a) 与手工网络相比:SLGE 网络提高了最流行的手工网络的性能。 SLGE 的分类误差分别比 ResNet-1001 和 FractalNet 低 0.8% 和 1.4%,参数很少。然而,DenseNet-BC (k=40) 比使用更多参数的 SLGE 网络表现更好。

b) 与增强NAS网络相比:表明SLGE对MetaQNN和NASv2分别实现了大约3.2%和2.3%的错误率提升;并与 Block-QNN-S（使用更多参数)和 NASNet-A（消耗更多 GPU 资源)竞争。通过 cutout augmentation 和更多的 1.8M 参数,ENAS 使用 0.5 GPU days 实现了比 SLGE 低 0.9% 的错误率。

c) 与基于梯度的 NAS 网络相比:使用 cutout augmentation 方法,DARTS 和 P-DARTS 使用较少
GPU天数实现分类错误率约为
分别比 SLGE 低 0.9% 和 1.2%。然而,SLGE 网络的参数少于 DARTS 和 P DARTS。 d) 与进化 NAS 网络相比:SLGE 的性能明显优于遗传 CNN 和大规模进化,但略落后于最先进的 Hier archical Evolution 和 AmeobaNet-A 网络。

SLGE 分别消耗了 Hierarchical Evolution 和 AmeobaNet-A 消耗的 0.06 和 0.006 GPU 计算日。

D. 随机搜索

为了评估 SLGE 表示方案的有效性,执行了一个简单的随机搜索作为基线。我们从顶部染色体随机生成了 10 个独立网络的群体（表 II）;在 CIFAR 10 上使用 500 个 epochs 的相同训练设置从头开始训练,没有任何进化修改。表 III 中报告了具有最低分类错误的网络。我们在十个单独的网络中实现了 5.85% 的平均分类误差和 1.77% 的标准偏差。

这是一个相当有竞争力的结果,证明了我们提出的具有常规卷积搜索单元的 SLGE 方案的有效性。

E. CIFAR-100 评估

在 CIFAR-10 任务上学习的顶级架构在 CIFAR-100 上进行了实验,以评估进化发现的细胞的可转移性。 CIFAR-100 具有与 CIFAR-10 相似的特性,但有 100 个类,这使得分类任务相当具有挑战性。我们只是转移

CIFAR-10 的最佳架构（表 II），但使用 100 个类别的分类器头从头开始训练，所有训练设置保持不变。我们实现了 MetaQNN 和 Genetic CNN 分别提高 4.1% 和 6.1% 的分类错误，并与其他网络竞争（表 III）。

F. 讨论

我们从初步的实验结果中做出一些观察。用 2 个基因的染色体开发的网络比用 3 个基因的染色体开发的网络表现更好，这是我们最意想不到的。顶部网络的进化发现的单元（图 3）具有类似于 DenseNet 的拓扑结构，具有格式良好的非对称卷积。

这强调了所提出方法的有效性，未来将进行进一步的实验以确定其稳健性。总的来说，CIFAR 10 和 CIFAR-100 分类任务的初步结果非常有希望，尽管没有调整进化参数。因此，自然的未来方向是调整进化参数，并将任务扩展到最具挑战性的 ImageNet 分类任务，以验证所提出方法的通用性。

五、结论

我们提出了一种有效的进化方法，该方法基于将 CE 元素嵌入 GEP 染色体的表示方案，发现高性能细胞作为 CNN 架构的构建块。我们表明，即使使用简单的随机搜索策略，我们的 SLGE 表示方案与常规卷积单元相结合也可以获得显着的结果。我们的顶级网络在 CIFAR-10 和 CIFAR-100 数据集上都获得了与最先进网络极具竞争力的性能。除了第 IV-F 节中的观察之外，我们将继续改进所提出的方法并将其扩展到其他视觉任务，例如语义分割和对象检测。

参考

[1] X. Yao，“进化的人工神经网络”，IEEE 会刊，卷。 87，没有。 9，第 1423–1447 页，1999 年。

[2] T. Elsken、JH Metzen 和 F. Hutter，“神经架构搜索：A 调查，”JMLR，卷。 20，没有。 55，第 1-21 页，2019 年。

[3] L. Xie 和 A. Yuille，IEEE ICCV 中的“Genetic CNN”。 IEEE，2017，页。 1388–1397。

[4] Y. Sun、B. Xue、M. Zhang 和 GG Yen，“基于块的完全自动化的 CNN 架构设计。” IEEE 神经网络和学习系统汇刊，第 1-13 页，2019 年。

[5] M. Suganuma、S. Shirakawa 和 T. Nagao，“设计卷积神经网络架构的遗传编程方法”，载于第 27 届 IJCAI 会议记录，2018 年，第 5369-5373 页。

[6] B. Zoph 和 QV Le，“使用强化学习进行神经架构搜索”，载于第 5 届国际学习表征会议论文集，2017 年。

[7] C. Ferreira，基因表达编程：人工智能的数学建模。施普林格，2006 年。

[8] Y. Sun、B. Xue、M. Zhang 和 GG Yen，“用于图像分类的进化深度卷积神经网络”，IEEE Transactions on Evolutionary Computation，卷。 24，没有。 2，第 394–407 页，2020 年。

[9] F. Assunc?ao、N. Lourenc?o、P. Machado 和 B. Ribeiro，“DENSER：深层进化网络结构化表示”，Genetic Prog。和进化机器，卷。 20，没有。 1，第 5-35 页，2019 年。

[10] JM Swafford、M. O Neill、M. Nicolau 和 A. Brabazon，“用语法进化中的模块探索语法修改”，遗传编程，2011 年，卷。 6621，第 310-321 页。

[11] F. Gruau，“使用细胞编码和遗传算法的神经网络合成”。博士论文，L universite Claude Bernard-lyon I，1994。

[12] T. Desell，“使用志愿计算的卷积神经网络的大规模演化”，Proc。 GECCO，2017 年，第 127-128 页。

[13] E. Real、A. Aggarwal、Y. Huang 和 QV Le，“图像分类器架构搜索的正则化演化”，Proc。美国人工智能协会，2019 年。

[14] E. Real、S. Moore、A. Selle、S. Saxena、YL Suematsu、J. Tan、QV Le 和 A. Kurakin，“图像分类器的大规模演变”，第 34 届 JMLR 会议记录，卷。 70，2017，pp. 2902–2911.

[15] H. Liu、K. Simonyan、O. Vinyals、C. Fernando 和 K. Kavukcuoglu，“Hierarchical representations for efficient architecture search”，第六届 ICLR 会议记录，2018 年。

[16] B. Zoph、V. Vasudevan、J. Shlens 和 QV Le，“学习可扩展图像识别的可迁移架构”，IEEE CVPR 会议论文集，2018 年，第 8697-8710 页。

[17] H. Pham、M. Guan、B. Zoph、Q. Le 和 J. Dean，“通过参数共享进行高效的神经架构搜索”，第 35 届 ICML 论文集，卷。 80，2018，p. 4095–4104。

[18] Z. Zhong、J. Yan、W. Wu、J. Shao 和 C.-L。 Liu，“实用的块式神经网络架构生成”，载于 IEEE CVPR 会议论文集，2018 年，第 2423-2432 页。

[19] J. Zhong、L. Feng 和 Y.-S。 Ong，“基因表达编程：一项调查”，IEEE Comp Intelligence 杂志，卷。 12，没有。 3，第 54–72 页，2017 年。

[20] X. Li、C. Zhou、W. Xiao 和 PC Nelson，“前缀基因表达编程”，GECCO 会议记录，2005，p. 25-31。

[21] F. Gruau，“优化和编译中的人工细胞开发”，Towards Evolvable Hardware，1996，vol. 1062，第 48-75 页。

[22] F. Gruau 和 K. Quatramaran，“交互式进化机器人的细胞编码”，CWI，阿姆斯特丹，Tech。众议员，1996 年。

[23] A. Krizhevsky、V. Nair 和 G. Hinton，“从微小图像中学习多层特征”，2009 年（2020 年 2 月 3 日访问）。 [在线的]。可用：https://www.cs.toronto.edu/ kriz/cifar.html [24] C. Ferreira，“问题解决中的基因表达编程”，在 Soft 比较。和行业：最近的应用，2002 年，第 635–653 页。

[25] K. He、X. Zhang、S. Ren 和 J. Sun，“用于图像识别的深度残差学习”，载于 IEEE 会议论文集。关于 CVPR，2016 年，第 770-778 页。

[26] K. He、X. Zhang、S. Ren 和 J. Sun，“深度残差网络中的身份映射”，ECCV 会议记录，2016 年，第 630-645 页。

[27] G. Larsson、M. Maire 和 G. Shakhnarovich，“Fractalnet：无残差的超深度神经网络”，ICLR 会议记录，2017 年。

[28] G. Huang、Z. Liu、L. vd Maaten 和 KQ Weinberger，“密集连接的卷积网络”，IEEE CVPR 会议论文集，2017 年，第 2261-2269 页。

[29] B. Baker、O. Gupta、N. Naik 和 R. Raskar，“使用强化学习设计神经网络架构”，第 5 届国际学习表征会议论文集，2017 年。

[30] H. Liu、K. Simonyan 和 Y. Yang，“DARTS：可微架构搜索”，国际学习表征会议论文集，2019 年。

[31] X. Chen、L. Xie、J. Wu 和 Q. Tian，“渐进式可微分架构搜索：弥合搜索和评估之间的深度差距”，国际计算机视觉会议论文集，2019 年，第. 1294–1303.

[32] LN Smith，“神经网络超参数的规范方法：第 1 部分 学习率、批量大小、动量和权重衰减”，2018 年，arXiv:1803.09820v2 [cs.LG]。