课题2：深度神经网络架构自动设计 研究报告

石依凡 中国人民大学

内容目录

内容结构

本研究报告从四个部分对当前的神经架构搜索领域部分研究成果展开说明。首先，我概括并整理了7篇论文的主要内容，并分析了这些研究成果的优势与局限性。其次，我对ENAS和DARTS两个轻量级的NAS方式进行了复现，分别搜索出了一个CNN Cell架构，并对论文中提到的架构进行了调优与验证。之后，我概括了架构搜索的流程，并把它抽象成流程图加以说明。最后，我对比了当前工作的权重共享方式，并从若干角度分析了它们的优势和局限性。

1 论文阅读

1.1 综述

在现在的神经架构搜索（Neural Architecture Search, NAS）领域，主要可以根据研究方法分为以下的几个类别：**基于强化学习的方法、基于梯度下降的方法和基于超网络的方法**。其中，基于梯度下降的方式DARTS虽然全局也使用了一个“超网络”，但因其超网络**仅仅代表一个模型的概率分布而不是所有模型的集合**，将其独立划分为一类。下面根据论文内容，对这几类方法的步骤和主要优势、局限性做出一定的抽象和概括。

- 基于**强化学习**的方法（**Reinforcement Learning NAS**）
  - **步骤**：使用强化学习的方法进行神经架构搜索，通过控制器产生一个架构结构，然后将这个结构的网络训练至收敛，之后根据反馈重新调整控制器模型，直至收敛
  - **主要优势**：准确，效果很好
  - **主要局限性**：消耗计算资源过大，无法适应大数据集，且难以判定是否到达最优
  - **代表工作**：*Neural Architecture Search with Reinforcement Learning (Zoph, et al.)*
- 基于**梯度下降**的方法（**Differentiable NAS**）
  - **步骤**：将原本离散的搜索空间变为连续的，即将抽象的架构编码为一组表示选择倾向性的向量，进而使用数学方法计算出梯度并重复更新这个向量和权重，最终同时得到最优架构和权重
  - **主要优势**：较为准确，计算速度快
  - **主要局限性**：搜索空间小，搜索出的架构简单，且容易过拟合，陷入局部最优
  - **代表工作**：*DARTS: Differentiable Architecture Search (Liu, et al.)*
- 基于**超网络**的方法（**One-Shot NAS**）
  - **步骤**：
    - 首先，构建并调优一个超网络（Supernet）或共享权重（Shared Weights）集合。每次从其中随机地根据某个策略（可以是强化学习给出的策略，也可以是均匀采样、Progressive Shrinking、Sandwich等方式）产生一个或若干架构，并通过一次“前向推理、后向传播”的过程更新该架构网络对应的共享权重，直至超网络收敛，此时超网络中所有模型均认为达到最优
    - 其次，使用某些算法（例如，进化算法、随机算法、层次筛选算法）根据一定的硬件限制条件从超网络中选取出一个最优的子结构，然后从头开始训练（Train From Scratch）或者直接使用（Without Post-processing），得到最终的最优模型
  - **主要优势**：准确，搜索空间大，（不考虑硬件适应性时）搜索速度快或（考虑硬件适应性时）适应性良好

- **主要局限性**：实现硬件适应性的同时Supernet难以训练，不实现硬件适应性时Supernet利用率低
- **代表工作**：*ENAS: Efficient Neural Architecture Search via Parameter Sharing (Pham, et al.)*；*Single Path One-Shot Neural Architecture Search with Uniform Sampling (Stamoulis, et al.)*；*Once-for-All: Train One Network and Specialize it for Efficient Deployment (Cai, et al.)*；*BigNAS: Scaling Up Neural Architecture Search with Big Single-Stage Models (Bender, et al.)*

本节的后续内容**分别**对7篇论文的**主要内容和优势、局限性**做出了概括和分析。

1.2 Neural Architecture Search with Reinforcement Learning

内容说明

这篇论文介绍了一种使用强化学习（RL）进行神经架构搜索（NAS）的方法。传统的架构设计方法是人工设计的，但是这种方法通常需要大量的前置经验和时间，因此效率不高，同时，虽然人们设计出来了很多种精巧的结构，但是由于很多部分是难以数学严密推理证明的，也很难得到真正的最优架构。本文提出了一种使用强化学习方法进行自动化架构搜索的方法。

这篇文章使用一个RNN网络作为控制器（Controller），可以生成一个RNN Cell的架构结构；然后将这种架构的模型训练至收敛，得出其在验证集上的准确性（accuracy）；然后将这个准确性作为反馈（reward），重新给控制器进行强化学习。

此方法可以产生一些简单的CNN网络架构，包含卷积核的相关特征、卷积操作的参数。如下图所示：

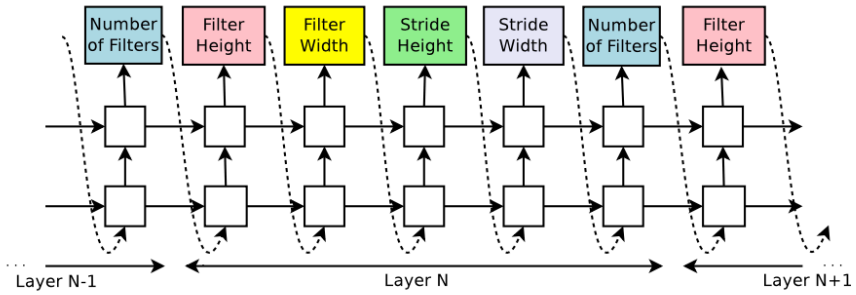


Figure 2: How our controller recurrent neural network samples a simple convolutional network. It predicts filter height, filter width, stride height, stride width, and number of filters for one layer and repeats. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.

因为需要使用**梯度更新参数**的方法，使用*Ronald J.Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In Machine Learning, 1992.*中提出的强化学习方法，并根据经验化简，然后使用数学技巧优化方差。最终得到Controller的参数梯度如下：

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_{\theta_c}\log P(a_t|a_{(t-1):1};\theta_c)(R_k-b)$$

同时，为了加速训练和搜索过程，这里使用了**分布式机器学习技术**。而为了增加产生架构的复杂度，提升搜索空间大小，控制器加入了**残差网络连接设计（Skip Connections）**，加入锚定节点（anchor）表明是否有这种连接方法。

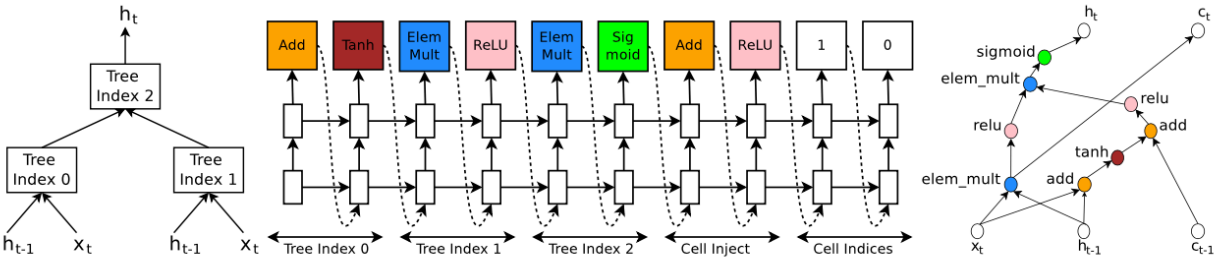


Figure 5: An example of a recurrent cell constructed from a tree that has two leaf nodes (base 2) and one internal node. Left: the tree that defines the computation steps to be predicted by controller. Center: an example set of predictions made by the controller for each computation step in the tree. Right: the computation graph of the recurrent cell constructed from example predictions of the controller.

上图是对从Controller的结果产生一个RNN Cell架构的例子。由一个k-叉树定义了总的架构，每一个树节点都有两个属性，分别表示预测的操作和激活函数。最后两个特殊的节点，完善了这个Cell结构。在论文的实现中，是一个8-叉树。

作者使用海量的计算资源（800GPUs）在CIFAR-10数据集上训练了12800个CNN架构，在Penn Treebank数据集上训练了35 epoches的RNN Cell架构，最终得到了很好的结果。

优势与局限性分析

这篇文章的**优势**在于：

- 使用强化学习的方式，**自动化**地对某一个CV问题设计出一套解决方案
- **架构相对多样**，搜索空间较丰富，可以是CNN，也可以是RNN Cell
- 取得了**很好的效果**

则篇文章的**局限性**在于：

- 效率低下，**对计算资源需求太大，无法进行大规模数据集训练**
- **CNN架构**仍然可以更好：没有考虑空洞卷积、分组卷积等方法
- **RNN架构**仍然可以更好：目前仅仅是针对Cell进行采样
- **数据集适应性不足**，在特定数据集上表现较好

1.3 DARTS: Differentiable Architecture Search

内容说明

本文使用**松弛化（Relaxation）**方法将离散的操作变成连续的权重向量最优化过程，最后再产生最终的架构。连续向量$\alpha = \alpha^{(i,j)}$实质上就是对这种架构的编码。因为是连续变量的最优化过程，可以你用梯度下降的方法优化模型，而不需要用到强化学习的方法。

具体的编码方式如下。使用softmax激活函数，将某一特定的操作变为所有可能操作的一个混合softmax。因此，可以使得用一组连续的权值向量来表示一个“**操作选择倾向**”，用$\overline{o}^{(i,j)}$表示第i和第j节点之间的操作倾向，而由于总操作集合$\mathcal{O}$已经提前确定，此倾向仅仅和连续向量$\alpha^{(i,j)}$有关。

$$\overline{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}$$

在产生具体的架构时，使用“**最可能的操作**”来代替这种“操作选择倾向”。即：

$$o^{(i,j)} = \underset{o \in \mathcal{O}}{\operatorname{argmax}}\ \alpha_o^{(i,j)}$$

然后将这个架构搜索问题转化为一个两次优化问题：**最小化架构编码向量给出的最终具体架构在验证集上的误差，具体架构由编码向量产生的架构在训练集上训练收敛得到。**使用**交替训练**（本身也是一种近似）的方式训练权重w和架构α，直至收敛。

随后，使用数学方法对架构编码向量的梯度进行近似。

- 首先，将最优情况下的权重时的编码向量的梯度**优化为一次优化后的权重对应的编码向量的梯度**
- 其次，将微分操作（由复杂的矩阵运算给出）**优化为一次有限差分**
- 【可选】：将二阶梯度直接考虑为0，**进行一阶近似**

最终，可以使得整体复杂度降低至可接受范围。

文章对CIFAR-10数据集进行了CNN搜索，对Penn Treebank数据集进行了RNN搜索，最终得到了比ENAS更好的CIFAR-10上的结果。同时，移植到了ImageNet等大型数据集上，证实此方法训练出的模型架构具有可移植性。

优势与局限性分析

本文的**优势**如下：

- 提出了一种新颖高效的方法，**将离散的搜索转为连续的搜索**
- **搜索空间更加灵活**，可以搜索任意形状和大小的子图
- 提出了**梯度计算的相关优化**方法
- **消耗的计算资源少**
- 搜索出的架构**表现好**

本文的**局限性**如下：

- 产生的**架构较为简单，搜索空间小**
- 由于搜索问题的非凸性、复杂性，**容易过拟合（陷入局部最优解），和随机种子关系较大**
- **消耗的计算资源仍不够少**（不如ENAS）
- 搜索空间需要提前指定（例如，操作总集合$\mathcal{O}$）

1.4 Efficient Neural Architecture Search via Parameter Sharing

内容说明

在这篇文章中，研究团队指出，传统的离散状态搜索的神经架构搜索（NAS）计算缓慢的主要原因在于“*The computational bottleneck of NAS is the training of each child model to convergence, only to measure its accuracy whilst throwing away all the trained weights.*”也就是说，**每次的权重参数训练有大量的冗余**。进而，研究团队提出了“**权值共享**”的策略，并以此设计了ENAS。ENAS能提升NAS的速度达1000倍，而结果可以与NAS相媲美。

首先，作者将整个搜索空间定义为一个完全的有向无环图（Complete DAG）$G = \{V, E\}$。每一个节点对应一个操作，而数据的流动作为其中的有向边。随后，在这个图中选取一定数量的边$E_0 \subseteq E$，得到一个子图$G_0 = G[E_0]$。这个子图便对应了神经架构搜索时一次采样出的一个架构。

控制器（Controller）仍然是一个RNN。在产生RNN Cell的架构时，对每个节点依次做出预测，决定下面的属性：

- 哪一条边作为输入
- 此节点对应的运算操作是什么

对于所有的没有输出边的节点，取他们的平均值作为这个Cell的输出。对于边$e \in E_0 : i \to j$来说，它代表的含义为：$h_j = \mathrm{Op}_j(h_i \cdot \mathbf{W}^{(\mathbf{h})}_{\mathbf{j,i}})$。这个$\mathbf{W}$便是**共享权重**，在整个网络训练的过程中，为所有模型共享。

在训练的过程中，主要是**交替的方式**训练Controller的参数θ和子网络的权重w。首先，固定控制器，使用蒙特卡洛方法**根据某一采样策略（由θ指出）**采样出多个子网络并计算出w的梯度（**并同步更新Supernet**）。在实验中发现，只需要采样出一个子网络计算出的梯度结果是可以接受的。随后，固定w，计算θ的梯度，同1.2部分中叙述的方法一致。最后，迭代多次直至收敛。此时，**可以认为超网络（Supernet）已经训练完毕**。

然后，**固定超网络相关参数，开始微调步骤**。采样出多个初始子模型，然后评估出最好的一个从头开始训练，得到最佳子模型和其权重。

与设计RNN架构类似，CNN架构搜索同样对每个节点做出两个决策：以之前哪些节点作为输入；使用什么样的操作。并且，本文还提出了一种搜索单个CNN Cell（Layer）来组成最终的CNN的方法，这种方法可以显著缩小之前CNN搜索的搜索空间。

ENAS的效果很好。在CIFAR-10和Penn Treebank数据集上测试，它比传统的NAS（1.2）只差一点，但是快1000倍。

优势与局限性分析

此研究的**优势**如下：

- **效率极高**，使用**权重共享**策略，比传统NAS快1000倍
- **效果好**，媲美传统NAS
- **搜索空间完全**，Supernet是一个完全图

此研究的**局限性**如下：

- 没有在**大规模数据集**（ImageNet）上运行效果测试
- **权重规模过高**，完全图的每一个边都有对应的权重矩阵
- **超网的利用程度不高**，每一个节点仅能多态出几种形态的架构
- **无法满足限制条件（Constraints）**

1.5 Searching for MobileNetV3

这篇文章侧重点不在神经架构搜索（NAS），仅仅是使用了修改权重因子来适应小模型的延时变化。因此，不在此详细阐述。

1.6 Single Path One-Shot Neural Architecture Search with Uniform Sampling

内容说明

基于梯度的方法容易**陷入局部最优解**，而忽略全局的最优解，跟起始选取的随机种子有很强的联系，目前仍然仍不清楚那些方式是有效的。对此，本文提出了一种使用**均匀采样策略**的NAS方法，通过随机的方式解决上述问题。并且，该方法采用"**Single Path One-Shot**"（单路径结构和一次性训练）的策略，大幅降低了搜索的成本。此方法在多个数据集上进行了实验，取得了媲美当下最优秀的NAS方法的结果。

最近的研究采用的"One-Shot"方法采用的是"Drop-Out"的方式产生路径，即随机去除一些Supernet中的边。这种方式可以均匀地训练到图中每一个参数，进而有望搜索到全局最优解。然而，这种方法对Drop-Out Rate特别敏感，从而使得整个训练特别困难。而本文使用的"Single Path"方式解决了这一问题。

与传统的Supernet不同，此文章提出的Supernet的训练目标是使得搜索空间中的所有架构被同时优化，也就是说，需得到一个最优的**共享权重**$W_{\mathcal{A}}$，使得这个Supernet根据架构a采样出的网络$\mathcal{N}(a, W(a))$的损失**期望**是最低的。也就是下面的公式所表达的含义：

$$W_{\mathcal{A}} = \underset{W}{\operatorname{argmin}} \underset{a \sim \Gamma(\mathcal{A})}{\mathbb{E}} [\mathcal{L}_{train}(\mathcal{N}(a, W(a)))]$$

搜索空间也被扩张了。文章提出了**通道数搜索和混合精度量化搜索**的策略，使得架构搜索可以适应多通道搜索和映射的位宽。这大大提升了搜索空间，增强了模型的复杂度。

最后，因为文章使用的是"One-Shot"方法，每次的架构都是随机生成的，没办法直接选出最优的架构进行训练，本文使用了**进化搜索**的策略，对每一个产生的网络$\mathcal{N}(a, W(a))$仅仅进行前向推理（Inference）得出其在验证集上的准确率，然后使用进化算法寻找最优的那个进行从头训练。

优势与局限性分析

此研究的**优势**在于：

- **效率高、性能好**
- **理论性能更好**，因为其通过一次训练、均匀采样等方式**降低了陷入局部最优的可能性**
- **搜索空间复杂度进一步提升**，可以进行多通道、多精度搜索
- 可以**满足一定的限制条件（Constraints）**
- 使用进化搜索的策略，改善搜索出的架构

此研究的**局限性**在于：

- **仅可以生成包含卷积层和全连接层的CNN网络**
- 采样策略是均匀采样，使得**仍然存在一定的范围限制**
- One-Shot策略会导致Supernet难以收敛，**Supernet训练至完全收敛的时长较高（相对于ENAS等交替训练的方法）**，多Epoches训练不能保证Supernet完全收敛

1.7 Once-for-All: Train One Network and Specialize it for Efficient Deployment

内容说明

普通的NAS方法对不同的设备上部署一个深度学习模型，都需要从头开始计算、搜索。这是非常大的一笔开销。本文提出了一种方法，使得Supernet只需要被训练一次（Once-For-All, OFA），就可以被用于很多个平台，进而做到高效地部署。在边缘设备上，OFA搜索出的模型有着更高的效率或是更高的准确率。

OFA试图产生的是CNN架构，它可以产生任意层（elastic depth）、任意通道数（elastic width）、任意卷积核大小（elastic kernel size）、任意图片大小（elastic resolution）的卷积神经网络。这极大的增大了搜索空间。

在训练Supernt的过程中，有两种基本思路：

- 在每一步，**枚举**出所有的子网，并精确计算梯度：**时间复杂度不可接受**
- 在每一步，**采样**出若干子网，并估算梯度：**很不准确且难以收敛**

本文提出了**Progressive Shrinking**的方法训练Supernet。首先训练大模型（Full Model），然后逐步**缩小不同的规模**（深度、宽度、核大小、分辨率），并**同时训练**大模型和缩小的模型（Shrinking Model），最终使得Supernet可以支持各种规模的模型。而大模型和小模型是**嵌套关系**，因此小模型**共享到了大模型的核心权值参数**（Weights），进而加速了训练过程。

对于深度、宽度和核大小的**权值共享策略**，文章分别做了说明。

- 不同大小的卷积核通过**变形**矩阵（Transformation Matrices）来共享权重
- 不同深度的模型通过**裁剪**出最先的D层参数用于共享
- 不同宽度的模型通过**排序**算法，找到最重要的若干个通道用于共享权重

最后，本文在训练出一个OFA Supernet后，测出了不同架构规模（Depth-Width-KernelSize）所对应的性能和效率指标（accuracy-latency twin）。也就是说，测算下面的映射关系，形成一张Look-Up Table，用于指导后续的使用该OFA Supernet进行的架构搜索工作：

$$\sigma : [D, W, K] \rightarrow [ACC, Latency]$$

优势与局限性分析

本研究有着**显著的优势**：

- 提出**无需重新训练**就能**灵活地支持不同的规模的参数（硬件条件）**的方法：OFA Supernet，训练Supernet的开销可以被后续诸多应用均摊
- 小模型可以直接从Supernet切出来，**无需或不怎么需要后处理**
- 提出了**Progressive Shrinking方法**，可以准确且迅速地训练Supernet
- **在不同的设备（和边缘设备）上进行了测试**，并且性能、效率指标都很好

本研究的**局限性**如下：

- Supernet对之后不同的应用场景来说**是一个黑箱**，如果训练完之后出了问题**难以解释、难以调优**
- Supernet是由一个Full Model进行训练的，之后所有的子model都用的是之前的权重继续训练，可能出现**陷入局部最优解**的情况（但是实践中仍取得了很好的效果）
- 测算出性能的Look-Up Table带来的**性能开销可能较长**，或者**需要牺牲掉大部分的搜索空间**

- Supernet**不能对不同的数据集和问题进行适配**，如果需要变换数据集或者问题，需要重新花费大量的资源进行训练
- Supernet的训练过程中不同规模的权重共享策略可能仍然有点不科学，无法做到理论上的无偏估计（但是实践中取得了很好的效果）

1.8 BigNAS: Scaling up Neural Architecture Search with Big Single-Stage Models

内容说明

传统的基于Supernet的NAS分为两个阶段：训练Supernet，并使用超大网络生成一个模型进一步调整（Post-Processing）。但是，这个后处理的过程可能会比较耗时。BigNAS的目标是提出一种**一步到位**的方式，直接生成可以使用的模型。

整体方法分为两步：

- 首先，**训练出一个规模庞大的超网络**，其可以直接通过切片等方式得到一个直接可部署的小模型
- 其次，在给定一些限制条件的时候，通过**粗力度到细粒度的筛选**，得到最优的模型。

区别于1.7所提到的Once-For-All方法，BigNAS使用**同时训练所有的规模的子模型**的方法。它每次训练时，使用**三明治方法**采样出最大的、最小的和N个随机大小的模型，计算出总的梯度并更新自身权重。

BigNAS使用了**原地蒸馏**（Inplace Distillation）的方式，使用Ground Truth计算最大的模型的损失，而使用最大模型预测出的Soft Label来计算其他模型的损失。这种方法保证了所有模型表达的一致性，可以略微增加效果。

同时，实验发现大模型和小模型的收敛速度不一致。大模型会过拟合而小模型仍未收敛。因此，将学习率换成**指数衰减+常量结尾（Exponentially Decaying with Constant Ending）**的方式，使得大模型在最优解附近摇摆，但是不会过拟合，而小模型会加速收敛。从而解决这个问题。

最后，需要选择出符合限制条件的最优模型。首先，给定若干**参照规模参数**值；然后，根据由此产生的模型，找到符合条件的最优规模参数；最后，**随机突变**这些参数，得到最终的最优模型。

优势和局限性分析

本研究和Once-For-All研究类似。

本研究的**优势**在于：

- 训练出一个Single-Stage的超网络，并可以**不经训练直接切出需要的模型**
- 使用优化方式，可以**同时训练出不同大小的最优模型**

本研究的**局限性**在于：

- BigNAS使用到了一定的超参数（如学习率常数），这些**超参数很难调整**
- 虽然BigNAS的目标是达成不经后处理即可直接产生最优模型，但是实践表明，可能在**某些场景下还是需要后训练**
- **粗力度到细粒度的筛选仍然需要一定时间的计算**，也可以认为是一种后训练

2 复现工作

我对每篇论文所开源出的仓库（若有）都进行了学习和研究，在分析每篇论文复现所需设备环境条件后，最终选择了**计算资源要求最低的ENAS、DARTS尝试复现**（复现服务器有使用时长限制）。

复现工作使用到的服务器环境如下：

- OS：Ubuntu 20.04.5 LTS x86_64
- CPU：12th Gen Intel i5-12600K (12) @ 3.600GHz
- GPU：NVIDIA GeForce RTX 2080 Ti Rev. A (12GB)
- Memory：32GB
- CUDA Version：10.0
- cuDNN Version：7.4.2

2.1 ENAS复现

在论文原文的Github仓库中，我拿到了研究团队公开的源代码。使用下面的环境进行复现：

- Python Version：2.7
- Tensorflow Version：1.13

我使用了和论文上完全一致的Tensorflow和Python版本，但是无法运行其对CIFAR-10数据集进行预设参数的神经架构搜索的代码，在多次试图修正后仍然失败，可能是服务器没有足够的内存（可能至少需要64GB内存）。因此，我修改了`data_util.py`中的预设参数，删除了CIFAR-10的第5个训练集batch，缩小了训练数据的数量。**重新运行自定义参数的搜索**，共搜索150个Epoches。

```
  train_files = [
    "data_batch_1",
    "data_batch_2",
    "data_batch_3",
    "data_batch_4",
    # "data_batch_5",    删除最后一个batch
  ]
```

在约5小时的训练后，我得出了下所示的结果。在验证集上的准确率为61.4%，在测试集上的准确率为59.75%。

```
[0 4 0 1 1 3 1 0 0 1 3 0 1 1 0 2 0 1 1 3]
[0 0 1 0 1 1 1 1 2 4 1 1 4 1 1 1 1 2 1 0]
val_acc=0.6250
--------------------------------------------------------------------------------
Epoch 150: Eval
Eval at 32850
valid_accuracy: 0.6140
Eval at 32850
test_accuracy: 0.5975
```

研究团队做出说明，一个具有`B + 2`个块的微型单元可以使用`B`个块来指定，这些块对应于编号为`2, 3, ..., B+1`的块，每个块由`4`个数字组成：

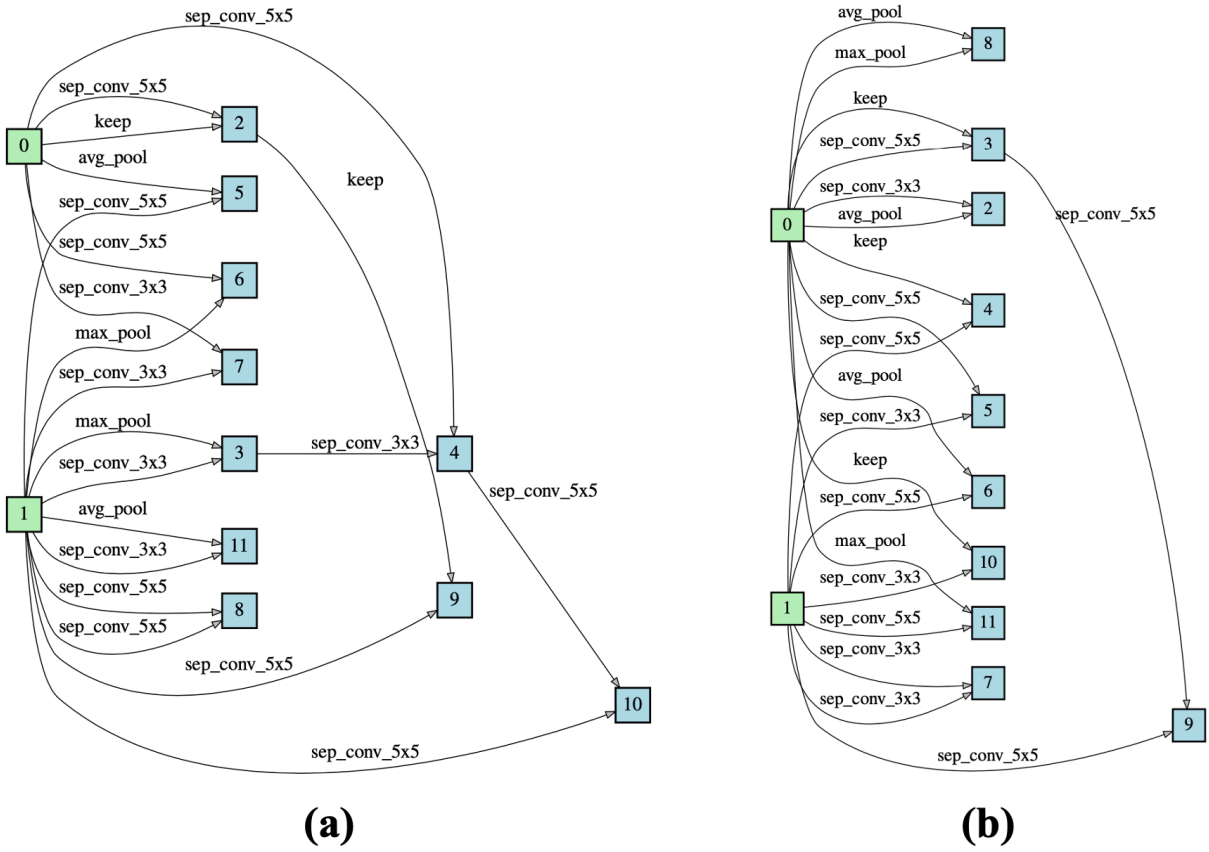
```
index_1, op_1, index_2, op_2
```

这里，`index_1` 和 `index_2` 可以是任何以前的索引。`op_1` 和 `op_2` 可以是 `[0, 1, 2, 3, 4]`，分别对应于 `可分离卷积_3x3`、`可分离卷积_5x5`、`平均池化`、`最大池化`、`恒等操作`。

研究团队同时也用类似的方式，给出了预搜索出的最优架构：

```
fixed_arc="0 2 0 0 0 4 0 1 0 4 1 1 1 0 0 1 0 2 1 1 1 0 1 0 0 3 0 2 1 1 3 1 1 0 0 4 0 3 1 1"
```

我使用画图工具，将我搜索到的和研究团队预先搜索好的CNN Cell分别可视化如下图**(a)**和**(b)**所示。可以看出，两个架构有一定的相似度，但因为我删除了部分训练集规模，并调整了部分参数，结果不尽相同。



最后，我对这个研究团队在论文中提到的预先使用`cifar-micro-search`搜索出的架构做了最终的调优。

```
epoch=629   ch_step=218900 loss=0.002567 lr=0.0001   |g|=0.5652   tr_acc=144/144
mins=1664.85
epoch=629   ch_step=218950 loss=0.013373 lr=0.0001   |g|=4.6466   tr_acc=143/144
mins=1665.21
epoch=629   ch_step=219000 loss=0.001038 lr=0.0001   |g|=0.1008   tr_acc=144/144
mins=1665.56
epoch=629   ch_step=219050 loss=0.001635 lr=0.0001   |g|=0.4037   tr_acc=144/144
mins=1665.92
epoch=629   ch_step=219100 loss=0.000772 lr=0.0001   |g|=0.1711   tr_acc=144/144
mins=1666.28
epoch=629   ch_step=219150 loss=0.002294 lr=0.0001   |g|=0.6714   tr_acc=144/144
mins=1666.63
epoch=629   ch_step=219200 loss=0.001445 lr=0.0001   |g|=0.5475   tr_acc=144/144
mins=1666.99
Epoch 630: Eval
Eval at 219240
test_accuracy: 0.9619
```

如上所示，在约30小时、630 Epoches的训练后，该模型达到了96.19%的准确率，即3.81%的分类误差，和论文中所描述的3.54%的分类误差（如下图所示）基本吻合。

Method	**GPUs**	**Times** (days)	**Params** (million)	**Error** (%)
DenseNet-BC (Huang et al., 2016)	–	–	25.6	3.46
DenseNet + Shake-Shake (Gastaldi, 2016)	–	–	26.2	2.86
DenseNet + CutOut (DeVries & Taylor, 2017)	–	–	26.2	**2.56**
Budgeted Super Nets (Veniat & Denoyer, 2017)	–	–	–	9.21
ConvFabrics (Saxena & Verbeek, 2016)	–	–	21.2	7.43
Macro NAS + Q-Learning (Baker et al., 2017a)	10	8-10	11.2	6.92
Net Transformation (Cai et al., 2018)	5	2	19.7	5.70
FractalNet (Larsson et al., 2017)	–	–	38.6	4.60
SMASH (Brock et al., 2018)	1	1.5	16.0	4.03
NAS (Zoph & Le, 2017)	800	21-28	7.1	4.47
NAS + more filters (Zoph & Le, 2017)	800	21-28	37.4	**3.65**
ENAS + macro search space	1	0.32	21.3	4.23
ENAS + macro search space + more channels	1	0.32	38.0	**3.87**
Hierarchical NAS (Liu et al., 2018)	200	1.5	61.3	3.63
Micro NAS + Q-Learning (Zhong et al., 2018)	32	3	–	3.60
Progressive NAS (Liu et al., 2017)	100	1.5	3.2	3.63
NASNet-A (Zoph et al., 2018)	450	3-4	3.3	3.41
NASNet-A + CutOut (Zoph et al., 2018)	450	3-4	3.3	**2.65**
ENAS + micro search space	1	0.45	4.6	3.54
ENAS + micro search space + CutOut	1	0.45	4.6	**2.89**

2.2 DARTS复现

在论文原文的Github仓库中，我拿到了研究团队公开的源代码。使用下面的环境进行复现：

- Python Version：3.6
- Pytorch Version：0.3.1
- Torchvision Version：0.2.0

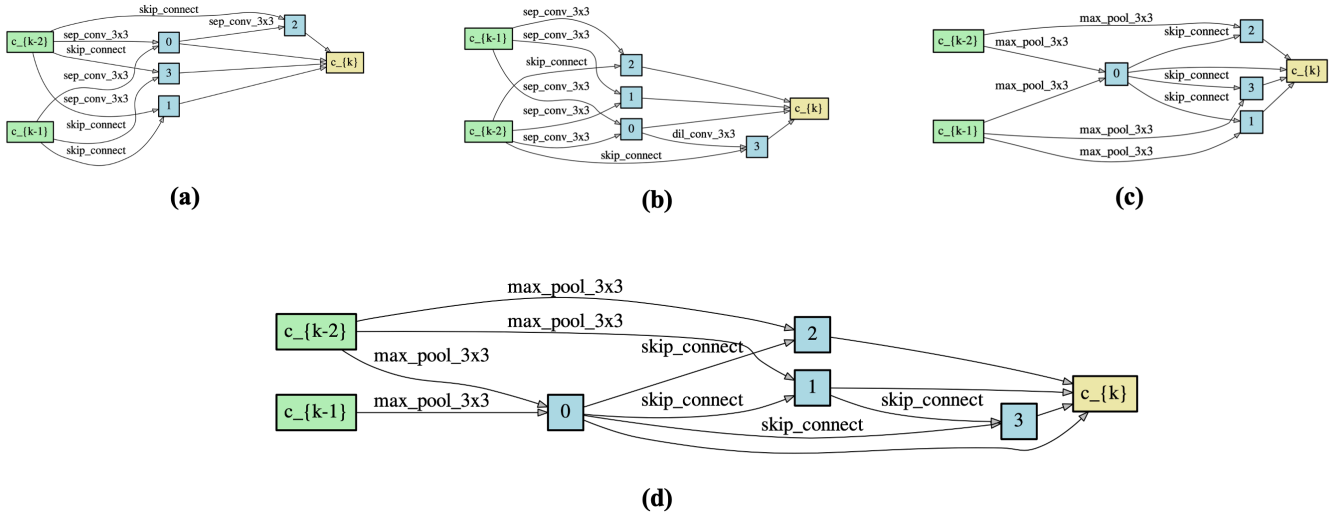
```
2023-04-03 08:37:30,882 train 000 4.491183e-02 98.437500 100.000000
2023-04-03 08:39:06,123 train 050 1.870327e-02 99.571078 100.000000
2023-04-03 08:40:41,354 train 100 1.855421e-02 99.644183 100.000000
2023-04-03 08:42:16,606 train 150 1.920131e-02 99.565397 100.000000
2023-04-03 08:43:51,818 train 200 1.885082e-02 99.611318 100.000000
2023-04-03 08:45:27,087 train 250 1.907752e-02 99.638944 100.000000
2023-04-03 08:47:02,286 train 300 1.911234e-02 99.652201 100.000000
2023-04-03 08:48:37,542 train 350 1.870171e-02 99.675036 100.000000
2023-04-03 08:49:53,315 train_acc 99.676000
2023-04-03 08:49:53,449 valid 000 4.773718e-01 87.500000 100.000000
2023-04-03 08:49:57,625 valid 050 3.327558e-01 90.318627 99.571078
2023-04-03 08:50:01,801 valid 100 3.695028e-01 89.402847 99.504950
2023-04-03 08:50:05,977 valid 150 3.672881e-01 89.372930 99.492964
2023-04-03 08:50:10,154 valid 200 3.796697e-01 89.132463 99.448072
2023-04-03 08:50:14,330 valid 250 3.788103e-01 89.162102 99.477092
2023-04-03 08:50:18,507 valid 300 3.811134e-01 89.192276 99.454942
2023-04-03 08:50:22,684 valid 350 3.814137e-01 89.271724 99.483618
2023-04-03 08:50:25,993 valid_acc 89.264000
```

首先，我在CIFAR-10数据集上使用论文中提到的**二阶近似**方法搜索了一个**层数为**8、**初始通道数为**16**的**模型，并搜索了50 Epoches，在约6小时的搜索后得到了上图所示的结果。此模型在训练集上有着非常优秀的表现，准确率高达99.676%，但是在验证集上准确率只有89.264%。这说明DARTS系列方法有着**过拟合**的风险。

该模型的具体参数如下：

```
genotype = Genotype(normal=[('sep_conv_3x3', 0), ('sep_conv_3x3', 1),
('sep_conv_3x3', 0), ('skip_connect', 1), ('skip_connect', 0), ('sep_conv_3x3', 2),
('skip_connect', 0), ('skip_connect', 1)], normal_concat=range(2, 6), reduce=
[('max_pool_3x3', 0), ('max_pool_3x3', 1), ('max_pool_3x3', 0), ('skip_connect', 2),
('skip_connect', 2), ('max_pool_3x3', 0), ('skip_connect', 3), ('skip_connect', 2)],
reduce_concat=range(2, 6))
```

使用研究团队提供的可视化工具，可将上述搜索到的CNN Cell架构和研究团队预先搜索到的架构可视化如下图所示。图**(a)**和图**(b)**分别展示了我和研究团队搜索出的Normal Cell，二者基本类似；图**(c)**和图**(d)**分别展示了研究团队和我搜索出的Reduction Cell。需要注意的是，由于运算过程包含随机操作，且我们的初始参数可能存在微小的不同，结果也存在着一定的差异。



随后，我运行了研究团队给出的预训练模型。在CIFAR-10的测试集上，它取得了97.37%的准确率，即2.63%的误差。这与论文所述的$2.76\% \pm 0.09\%$相一致。

```
Files already downloaded and verified
04/03 11:56:30 PM test 000 1.233736e-01 96.875000 100.000000
04/03 11:56:34 PM test 050 1.105459e-01 97.120095 99.959150
04/03 11:56:37 PM test 100 1.074739e-01 97.359733 99.948432
04/03 11:56:37 PM test_acc 97.369997
```

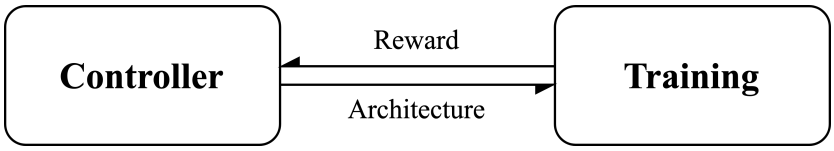
Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	#ops	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	–	–	manual
NASNet-A + cutout (Zoph et al., 2018)	2.65	3.3	2000	13	RL
NASNet-A + cutout (Zoph et al., 2018)†	2.83	3.1	2000	13	RL
BlockQNN (Zhong et al., 2018)	3.54	39.8	96	8	RL
AmoebaNet-A (Real et al., 2018)	3.34 ± 0.06	3.2	3150	19	evolution
AmoebaNet-A + cutout (Real et al., 2018)†	3.12	3.1	3150	19	evolution
AmoebaNet-B + cutout (Real et al., 2018)	2.55 ± 0.05	2.8	3150	19	evolution
Hierarchical evolution (Liu et al., 2018b)	3.75 ± 0.12	15.7	300	6	evolution
PNAS (Liu et al., 2018a)	3.41 ± 0.09	3.2	225	8	SMBO
ENAS + cutout (Pham et al., 2018b)	2.89	4.6	0.5	6	RL
ENAS + cutout (Pham et al., 2018b)*	2.91	4.2	4	6	RL
Random search baseline‡ + cutout	3.29 ± 0.15	3.2	4	7	random
DARTS (first order) + cutout	3.00 ± 0.14	3.3	1.5	7	gradient-based
DARTS (second order) + cutout	2.76 ± 0.09	3.3	4	7	gradient-based

* Obtained by repeating ENAS for 8 times using the code publicly released by the authors. The cell for final evaluation is chosen according to the same selection protocol as for DARTS.
† Obtained by training the corresponding architectures using our setup.
‡ Best architecture among 24 samples according to the validation error after 100 training epochs.

3 架构搜索流程概述

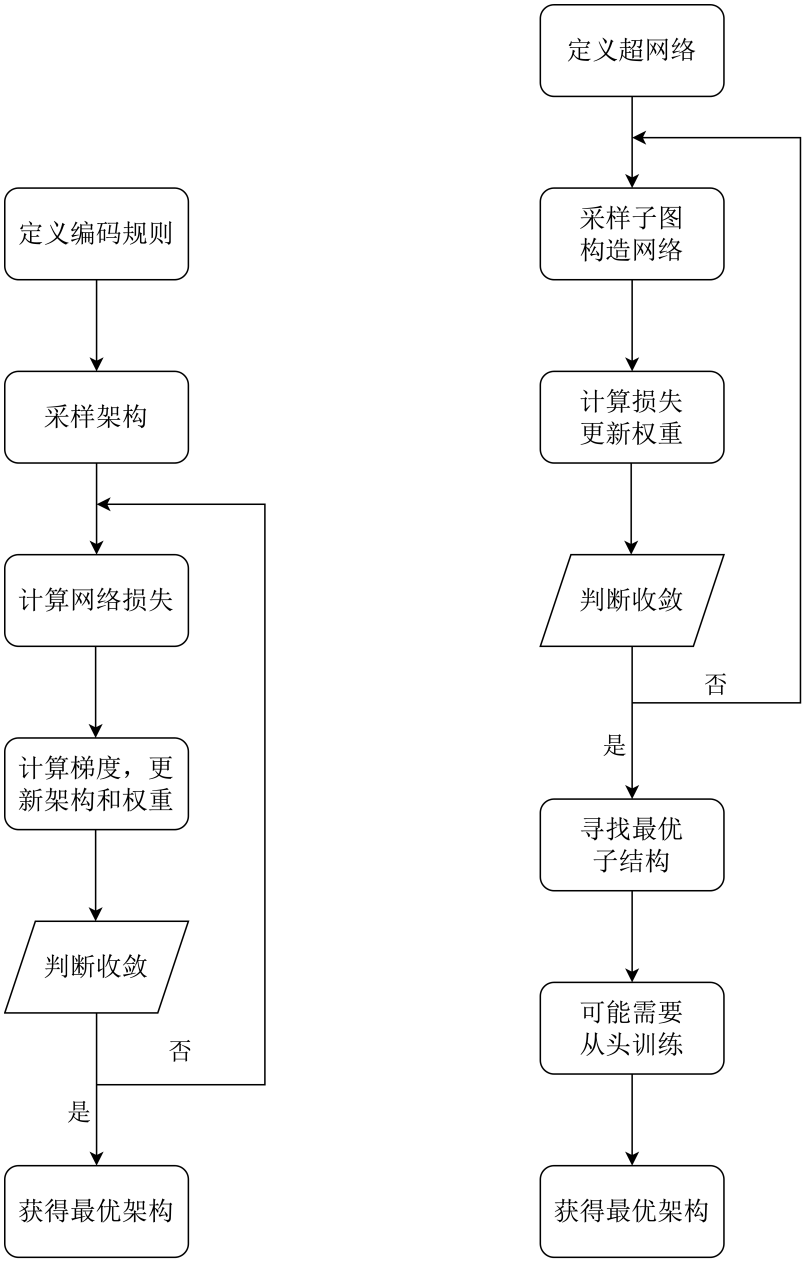
对于`1.1 综述`中所提到的三种类别的NAS（**基于强化学习的方法、基于梯度下降的方法和基于超网络的方法**），它们有着各自特有的流程。下面对其分别用流程图的方式加以说明。

3.1 基于强化学习的方法



对于基于强化学习的方法，NAS的过程较为简洁，主要就在一个不断迭代的过程中：控制器（Controller）根据策略产生架构并训练至收敛；然后评估模型，向控制器发回反馈（Reward）并更新生成策略；最终收敛，得到最优模型。

3.2 基于梯度下降的方法



上图**(a)**展示了基于梯度下降的方法的工作流程：

1. 将原本离散的搜索空间变为连续的，即将抽象的架构编码为一组表示选择倾向性的向量

2. 采样出一个编码α（即一个分布情况），并根据编码得到该分布最接近的初始网络$\mathcal{N}$（重新离散化）
3. 计算该网络的损失$\mathcal{L}$和梯度（∇_w和∇_α），分别更新权重w和架构编码向量α
4. 如果没有收敛，重复上一步
5. 得到最优架构

3.3 基于超网络的方法

上图**(b)**展示了基于超网络的方法的工作流程，可以被分为**训练超网络**（1～4）和**寻找最优架构**（5～7）两个阶段：

1. 定义一个超网络，一般由一个有向无环图G表示，代表整个搜索空间
2. 根据某种策略（随机或者根据上一轮结果）采样一个（或若干个）子网络$G_0 \subset G$，并由此形成一个（或若干个）架构α
3. 计算此架构的损失，并更新G中G_0部分的权重
4. 如果没有收敛，重复前两步，最终得到训练好的超网络
5. 根据某种算法，寻找最优的、适应限制条件的子结构
6. 可能还需要根据实际情况对子结构进行从头训练
7. 得到最优架构

4 权重共享方式分析

这几篇文章中，涉及到权重共享方式的有：

- **DARTS**：*DARTS: Differentiable Architecture Search (Liu, et al.)*
- **ENAS**：*ENAS: Efficient Neural Architecture Search via Parameter Sharing (Pham, et al.)*
- **SPOS**：*Single Path One-Shot Neural Architecture Search with Uniform Sampling (Stamoulis, et al.)*
- **OFA**：*Once-for-All: Train One Network and Specialize it for Efficient Deployment (Cai, et al.)*
- **BigNAS**：*BigNAS: Scaling Up Neural Architecture Search with Big Single-Stage Models (Bender, et al.)*

根据第一节的说明与概括，我得出了下面的结论。在本节之后的说明中，将使用上面加粗的名称（DARTS, ENAS, SPOS, OFA, BigNAS）指代上述论文的研究成果。同时，为对比的合理性，我将统一对比其**CNN网络架构的搜索方式**的差异。

4.1 权重共享方式及搜索空间情况

DARTS提出的梯度下降方法（通过连续化松弛方法，Continuous Relaxation）是一个*广义的权重共享方式*。它使用的超网络表达的是一个模型的分布情况，而不是所有的模型的母集，不能切出多个子网络。因为这个超网络只有一次架构初始化过程，全程使用的是同一个模型，没有重新采样步骤，因此其权重为每一轮迭代的子架构所共享。搜索空间不大，需要提前给定所有可能的情况作为预置空间。

ENAS使用一系列的权重矩阵$\mathbf{W}_{\mathbf{j,i}}^{(\mathbf{h})}$来表示超网络$G$中从节点$i$流向节点$j$的数据所乘的权重。该方式的搜索空间为预设的一个超网络G，每次控制器（Controller）通过一定的策略θ采样一个子图$G_0 \subset G$，形成本次迭代的架构α，然后根据对网络的具体损失$\mathcal{L}_{train}(\mathcal{N}(\alpha, w))$更新权重$w$和策略$\theta$。

SPOS的超网络是一个庞大的分层结构，其每一个节点（选择块，Choice Block）包含了多种同样结构的操作的选择，在整个Supernet中的一条路径（Single Path）代表着一个最终的架构。该方法的每一个操作选择对应了一个权重，该权重为Supernet采样出的所有架构所共享。其搜索空间很大，每一个节点可以有多种采样选择，每一个卷积层也可以有多种通道数、量化精度选择。

OFA的超网络是一个更加庞大的分层结构，其每一个节点都有深度、宽度、卷积核大小等不同的选择，对每一个节点都有一个共享的权重。采样的策略是Progressive Shrinking，逐步向更小规模的架构训练，更小的架构的某一节点会使用更大的架构的这一节点的一部分权重。

BigNAS的超网络类似于**OFA**，但是更换采样策略为Sandwich，每次都使用若干的大小不一的架构进行训练。同时，BigNAS的目标更侧重于不经额外训练直接得到最终模型。

4.2 优势与局限性对比

搜索空间情况

如上节所说，在Supernet大小相似的情况下，五种方法的搜索空间大小排序如下：

$$\mathbf{DARTS} \approx \mathbf{ENAS} < \mathbf{SPOS} < \mathbf{OFA} \approx \mathbf{BigNAS}$$

- **DARTS和ENAS**每个权重仅仅被预先定义的情况共享
- **SPOS**每个权重被机器随机生成的若干个通道数情况、量化精度情况共享
- **OFA和BigNAS**每个权重可以被切割成不同的深度、宽度、卷积核大小选择对应的权重

因此，每一个权重可变异多态数逐步增大，同时使得搜索空间逐步增大（**更偏向于微观的硬件环境支持增多，而不是架构宏观变大、复杂度增加**）。

准确度情况

在准确度和搜索空间的Trade-Off后，单个权重的可变异多态数也不是越高越好的。对硬件环境多态的偏好将**有损单个权重的表达能力**，会造成权重的收敛更加困难，且可能偏离对某一特定硬件条件下的最优值。

因此，在后面的研究中，想达到同样的效果，会造成Supernet规模扩张、训练时长严重加长（尤其是**OFA和BigNAS**）。但因其是“Once-For-All”的，或可以接受众多的Constraints，这也是一个合理的代价。

计算时间消耗情况

DARTS、**ENAS和SPOS**的超网络设计初衷是为了**通过共享权重的方式加速计算**，因此其非常高效，但是对不同的限制条件的支持非常弱或根本没有支持。

而**OFA和BigNAS**设计初衷是为了**实现一个可以重复用于多硬件条件的超网络**来避免重复计算，因此单个超网络的计算成本虽然高昂，但可以被之后的重复使用给均摊。

内存消耗情况

DARTS的超网络因只表达了一个模型，因此，在运算时需要把整个网络的参数全部读取进入内存（显存）中，这在Supernet规模相似的情况下是不利的。但是，**DARTS**的Supernet规模正常情况下是不会过大的。

ENAS和SPOS因同一时间仅有一个被采样出的子图在训练，因此，内存开销很小。

OFA和BigNAS因同一时间有若干个被采样出的子图在同时训练，因此内存开销可能较大。但考虑到训练这样的Supernet的设备应当都是强大的，因此不是主要局限性。

超参数敏感性

DARTS对超参数较为敏感，因为使用的是梯度下降的方法，只能找到局部最优解。初始架构的选择至关重要。

ENAS和BigNAS都有对结果影响较大的超参数，如预先定义的搜索空间、学习率常量等。

SPOS和OFA的过程随机性较高，根据论文描述，未使用到较重要的超参数。

数学可解释性

DARTS使用的方法有较完备的数学基础，保证了其一定可以在合适的参数情况下，找出局部最优解。

其他的方法难以解释为什么共享权重是有效的。

小结

在对本课题进行研究后，我对NAS的基本方法有了初步了解。通过阅读代码、尝试复现的方式，并在对多个方向的NAS工作进行分析后，我也对整个NAS的发展历史和未来发展趋势有了进一步的认识。

简单而言，NAS搜索正从一开始的“**单一模型、资源消耗大、解释性弱、敏感程度高**”向着“**多硬件条件、平均资源消耗小、解释性强、稳定性强**”发展，这个线索也将继续指导着后续的研究。

参考文献

[1] Cai H, Gan C, Wang T, et al. Once-for-all: Train one network and specialize it for efficient deployment[J]. arXiv preprint arXiv:1908.09791, 2019.

[2] Cai H, Zhu L, Han S. Proxylessnas: Direct neural architecture search on target task and hardware[J]. arXiv preprint arXiv:1812.00332, 2018.

[3] Guo Z, Zhang X, Mu H, et al. Single path one-shot neural architecture search with uniform sampling[C]//Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16. Springer International Publishing, 2020: 544-560.

[4] Howard A, Sandler M, Chu G, et al. Searching for mobilenetv3[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 1314-1324.

[5] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images[J]. 2009.

[6] Liu H, Simonyan K, Yang Y. Darts: Differentiable architecture search[J]. arXiv preprint arXiv:1806.09055, 2018.

[7] Pham H, Guan M, Zoph B, et al. Efficient neural architecture search via parameters sharing[C]//International conference on machine learning. PMLR, 2018: 4095-4104.

[8] Tan M, Chen B, Pang R, et al. Mnasnet: Platform-aware neural architecture search for mobile[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 2820-2828.

[9] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Reinforcement learning, 1992: 5-32.

[10] Yu J, Jin P, Liu H, et al. Bignas: Scaling up neural architecture search with big single-stage models[C]//Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16. Springer International Publishing, 2020: 702-717.

[11] Zoph B, Le Q V. Neural architecture search with reinforcement learning[J]. arXiv preprint arXiv:1611.01578, 2016.