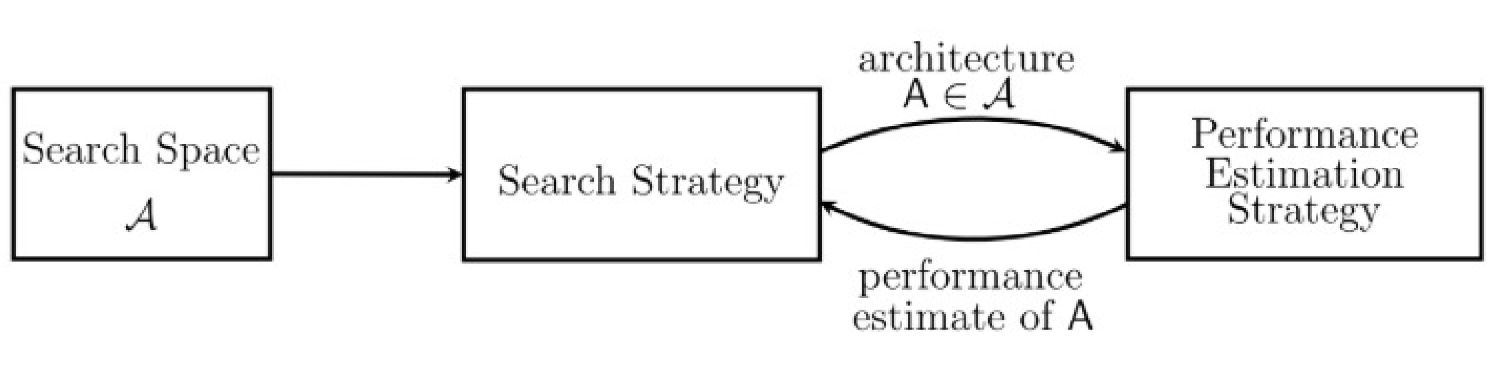
神经架构搜索

0.简介

深度学习模型在很多任务上都取得了不错的效果，开发合适的神经网络模型通常需要大量的架构工程，有时可以通过迁移学习获得，但如果真的想要获得最优性能，通常最好设计网络。但这需要专业技能（从商业角度看是昂贵的）并且总的来说具有挑战性，这是一个需要很多试验和错误（trial and error）的工作，实验本身是耗时且昂贵的。神经架构搜索（NAS）是一种自动设计神经网络的技术，可以通过算法根据样本集自动设计出高性能的网络结构，在某些任务上甚至可以媲美人类专家的水准，甚至发现某些人类之前未曾提出的网络结构，可以有效的降低神经网络的使用和实现成本。

NAS基本包括三个部分：搜索空间，搜索策略和性能评估策略。

NAS的原理是给定一个称为搜索空间的候选神经网络结构集合，用某种策略从中搜索出最优网络结构。神经网络结构的优劣即性能用某些指标如精度、速度来度量，称为性能评估。这一过程如下图所示。在搜索过程的每次迭代中，从搜索空间产生“样本”即得到一个神经网络结构，称为“子网络”。在训练样本集上训练子网络，然后在验证集上评估其性能。逐步优化网络结构，直至找到最优的子网络。



1. 组成部分

1.1 搜索空间

搜索空间定义了可以搜索的神经网络结构的集合，即解的空间。搜索空间的设计可以结合任务属性的先验知识，从而减小搜索空间大小并简化搜索。然而，这也引入了人为偏见，可能会阻止找到超越当前人类知识的新颖架构构建块。

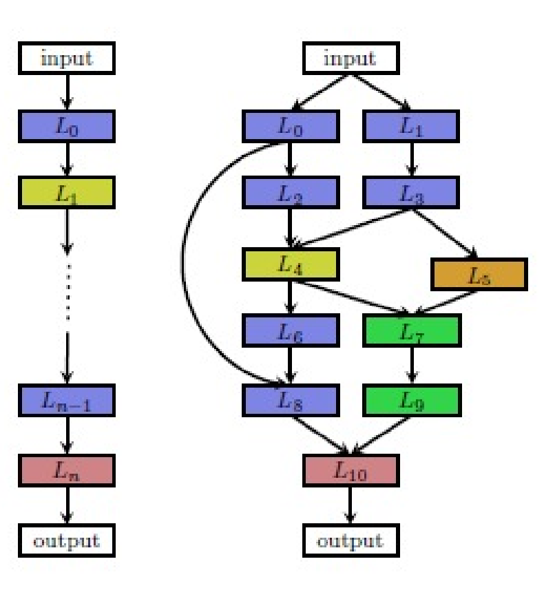
搜索空间根据网络类型可以划分为链式架构空间、多分支架构空间、cell/block构建的搜索空间。根据搜索空间覆盖范围可分为 macro（整个网络架构进行搜索）和 micro（仅搜索cell，根据cell扩展搜索空间）。

1.1.1 链式神经网络

如下图左侧，展示了一个相对简单的搜索空间，称为链式神经网络，直接将前一层的输入作为后一层的输出 。针对于这样一个 search space，需要考虑如下参数：

- 网络的层数 n
- 每一层执行的操作类型，比如池化。卷积这样的基础操作，或者更高级的一些操作，如深度可分离卷积、空洞卷积等。
- 每一层与这个操作相关的超参数，比如对于一个一般的卷积层来说，有卷积核个数、步长、尺寸等。

与每层相关的超参数取决于这一层的操作类型的，因此对于 Search Space 的参数化的结果并不是一个固定的长度，而是一个条件空间（conditioanl space），也就构成了搜索空间。



1.1.2 cell

除了简单的链式神经网路外，随之额NAS技术的完善，还引入了人工设计出的如跳跃连接(skip connections)等架构元素和模块单元，用来构建上图右侧所示的复杂的多分支网络。这样的方式明显的提高了架构的自由度，受其影响，也提出将NAS搜索整个架构转变为搜索这样的模块单元，这些模块称为 cells 或 blocks。相比于之前，这样的 search space 的优势:

1. 搜索空间的大小显著减小，将大量基础操作类型封装为固定的模块，提高搜索效率；
2. 仅仅通过调整一个模型中使用的 cells 的数量，就可以很容易地将应用到其他的数据集。

因此，这种 cell-based 的 search space 也应用在了很多后来的研究中。然而，当使用基于 cell 的搜索空间时，出现了新的选择问题，即如何选择元架构（meta-architecture）：即应该使用多少 cells 以及如何连接它们来构建实际模型。这部分的工作也是当前的研究热点之一。如在 meta-architecture 的优化方面，有工作提出了一种分层搜索空间（hierarchical search space）的方法将搜索分成了几个不同 level，逐级连接搜索。

Search space 的选择很大程度上就决定了这一优化问题的难度：即使对于基于单个 cell 且 meta-architecture 固定了的这样的搜索空间情况而言，挑战仍然还包括不连续性和过高维度的问题。

1.2 搜索策略

搜索策略定义了如何在搜索空间中寻找最优网络结构。搜索算法通常是一个迭代过程，定义了使用怎样的算法可以快速、准确找到最优的网络结构参数配置。常见的搜索方法包括：随机搜索、贝叶斯优化、进化算法、强化学习、基于梯度的算法。其中强化学习和进化学习是主流算法。

在搜索过程的每个步骤或迭代中，从搜索空间产生“样本”形成一个神经网络，称为“子网络”。所有子网络都在训练数据集上进行训练，然后将它们在验证数据集上的准确性视为目标（或作为强化学习中的奖励） 进行优化。搜索算法的目标是找到优化目标的最佳子网络，例如最小化验证损失或最大化奖励。

- 基于强化学习的搜索方法

创性的工作主要是2016年由MIT发表的《Designing Neural Network Architectures using Reinforcement Learning》和Google发表的《Neural Architecture Search with Reinforcement Learning》两篇文章。前者提出MetaQNN，它将网络架构搜索建模成马尔可夫决策过程，使用RL方法（具体是Q-learning算法）来产生CNN架构。对于CNN的每一层，学习体会选取层的类型和相应参数。生成网络结构后训练后得到的评估精度作为回报。这个回报用以参于Q-learning训练。后者采用RNN网络作为控制器来采样生成描述网络结构的字符串，该结构会用于训练并得到评估的准确率，然后使用了REINFORCE算法（早期的RL方法，在他们后面的工作中又使用了比较新的更加sample efficient的PPO算法）学习控制器的参数，使之能产生更高准确率的网络结构。

基于强化学习(reinforcement learning ,RL)的方法已经成为NAS的主流方法。RL有四个基元：agent, action,environment和reward. 强化学习是通过奖励或惩罚(reward)来学习怎样选择能产生最大积累奖励的行动(action)的算法。

在 NAS 任务中，将架构的生成看成是一个 agent 在选择 action，reward 是通过一个测试集上的效果预测函数来获得（这个函数类似于工程优化问题中的 surrogate model，即代理模型），agent 与环境没有交互，可以降阶为无状态的多臂老虎机（MAB）问题。这类工作整体的框架都是基于此，不同的点在于策略表示和优化算法。

核心思想是通过一个controller RNN在搜索空间中得到一个子网络结构，然后用这个子网络结构在数据集上训练，在验证集上测试得到准确率，再将这个准确率回传给controller，controller继续优化得到另一个网络结构，如此反复进行直到得到最佳的结果。

在探索高维搜索空间时，基于RL的搜索成本非常高。

- 进化算法

进化算法是一大类算法，大概的框架也基本类似，先随机生成一个种群（N 组解），开始循环以下几个步骤：选择、交叉、变异，直到满足最终条件。在一个进化过程中，工作者（worker）随机从模型簇中选出两个个体模型；根据优胜劣汰对模型进行识别,不合适的模型会立刻从模型簇中被剔除，即代表该模型在此次进化中的消亡；而更优的模型则成为母体（parent model），进行繁殖；通过这一过程，工作者实际上是创造了一个母体的副本，并让该副本随机发生变异。研究人员把这一修改过的副本称为子代（child）；子代创造出来后，经过训练并在校验集上对它进行评估之后，把子代放回到模型簇中。此时，该子代则成为母体继续进行上述几个步骤的进化。

在Google的论文《Large-Scale Evolution of Image Classifiers》中，进化算法被引入来解决NAS问题。首先，网络结构会进行编码，称为DNA。演进过程中会维护网络模型的集合，这些网络模型的fitness通过它们在验证集上的准确率给出。在进行过程中，会随机选取两个模型，差的那个直接被淘汰，好的那个会成为父节点。子节点经过变异（在一些预定的网络结构变更操作中随机选取）形成子结点。子结点经过训练和验证过放入集合。该作者在后续论文《Regularized Evolution for Image Classifier Architecture Search》中，提出了该方法的变体aging evolution，让进化中的选择倾向于比较『年轻』的模型，它可以帮助更好地进行探索。经搜索出的最优网络结构称为AmoebaNet。

另外作者对强化学习，进化算法和随机搜索作了比较，发现强化学习和进化算法从准确率上来说表现很好。与强化学习相比进化算法搜索得更快（尤其是早期），且能得到更小的模型。随机搜索在学习到的网络准确率上会次于前面两者，但差距不是那么大（尤其是小数据集上）。

进化算法是一种无梯度的优化算法（Derivative Free Optimization Algorithm），优点是可能会得到全局最优解，缺点是效率相对较低，近两年几家高科技企业做 NAS 时都在用进化算法优化网络结构，同时用基于梯度的方法（BP）来优化权值。在 NAS 任务中，进化算法的交叉算子和任务结合比较紧，被定义为一些类似添加、删除层的操作，而非简单的更改某一位编码。

用进化算法解决 NAS 问题，不同的工作可能聚焦在不同的过程中，比如如何 sample 种群，如何 update 种群，如何生成子代种群等。进化学习的一个缺点是进化过程通常不稳定，最终的模型簇质量取决于随机变异。后有学者提出通过RL 控制器确定变异替代随机变异，稳定搜索过程。

- 贝叶斯优化

贝叶斯优化（Bayesian Optimization）是超参数优化问题的常用手段，尤其是针对一些低维的问题。对于高维优化问题，有相关工作融合了树模型或者随机森林来解决，取得了不错的效果。如2018年的**Auto-Keras**提出利用贝叶斯优化(BO)算法，通过神经架构渐变来探索搜索空间，并在此工作的基础上构建Auto-Keras开源AutoML系统。

- 基于梯度的方法

这是比较新的一类方法。前面的基于强化学习和进化算法的方法本质上都还是在离散空间中搜索，它们将目标函数看作黑盒。但如果搜索空间连续，目标函数可微，那么基于梯度信息可以更有效地搜索。CMU和Google的学者在《DARTS: Differentiable Architecture Search》一文中提出DARTS方法。一个要搜索最优结构的cell，可以看作是包含N个有序结点的有向无环图。结点代表隐式表征（例如特征图），连接结点的的有向边代表算子操作。DARTS方法中最关键操作是将候选操作使用softmax函数进行混合。这样就将搜索空间变成了连续空间，目标函数成为了可微函数。这样就可以用基于梯度的优化方法找寻最优结构。搜索结束后，这些混合的操作会被权重最大的操作替代，形成最终的结果网络。另外，中科大和微软发表的论文《Neural Architecture Optimization》中提出另一种基于梯度的方法。它的做法是先将网络结构做嵌入（embedding）到一个连续的空间，这个空间中的每一个点对应一个网络结构。在这个空间上可以定义准确率的预测函数。以它为目标函数进行基于梯度的优化，找到更优网络结构的嵌入表征。优化完成后，再将这个嵌入表征映射回网络结构。这类方法的优点之一就是搜索效率高。

1.3 性能评估策略

由于深度学习模型的效果非常依赖于训练数据的规模，大规模数据上的模型训练会非常耗时，对优化结果的评价将会非常耗时，所以需要一些手段去做近似的评估。

一种思路是用一些低保真的训练集来训练模型，比如训练更少的次数，用原始训练数据的一部分，低分辨率的图片，每一层用更少的滤波器等。用这种低保真的训练集来测试优化算法会大大降低计算时间，但也存在一定的偏差。

另一种主流思路是借鉴于工程优化中的代理模型，在很多工程优化问题中，每一次优化得到的结果需要经过实验或者高保真仿真（有限元分析）进行评价，实验和仿真的时间非常久，不可能无限制地进行评价尝试，学者们提出了一种叫做代理模型的回归模型，用观测到的点进行插值预测，这类方法中最重要的是在大搜索空间中如何选择尽量少的点预测出最优结果的位置。

第三种主流思路是参数级别的迁移，用之前已经训练好的模型权重参数对目标问题进行赋值，从一个高起点的初值开始寻优将会大大地提高效率。在这类问题中，积累了大量的历史寻优数据，对新问题的寻优将会起到很大的帮助，用迁移学习进行求解，也是不错的思路。

第四种思路叫做单次（One-Shot）架构搜索，这种方法将所有架构视作一个 one-shot 模型（超图）的子图，子图之间通过超图的边来共享权重。

2. 最新进展和方向

最新的进展集中于以下方面：

1. 将NAS的思想用到不同领域或对象上。如对多尺度特征融合方法进行搜索的NAS-FPN，用于目标检测的数据增强方式的组合搜索。
2. 计算消耗成本的节约，如ENAS等方法，不断减少训练所需要的计算成本，实现易操作和通用性，克服对大量计算设备的依赖
3. 搜索策略的优化。通过优化搜索策略不仅可以减少计算消耗，而且能取得更好的效果，如IRLAS

   、RENAS等。

2.1 搜索加速

上面提到，由于NAS中涉及的搜索空间巨大，而且其性能评估往往得涉及模型的训练，导致消耗的资源很大。像前面提到的基于强化学习和进化算法的方法，对于CIFAR这样不算大的数据集，基本都是用了上千GPU/天。因此，有很多工作着手解决这个问题。其中比较典型的加速方法有：

- 层次化表示(Hierarchical Representation)：如果要对一整个神经网络结构进行搜索，搜索空间是非常大的。Google的论文《Learning Transferable Architectures for Scalable Image Recognition》提出NASNet，它假设整体网络是由cell重复构建的，那搜索空间就缩小到对两类cell（normal cell和reduction cell）结构的搜索上，从而大大减小了搜索空间。在这个方法中，虽然cell结构是学习得到的，但如何重复和组合这些cell的元网络结构是预定义的。更进一步，CMU和Google发表的论文《Hierarchical Representations for Efficient Architecture Search》中定义了一种层次化的网络结构：最底层为像卷积和池化等基本组件；中间层为这些组件所构成的图；最高层就是由这些图层叠而成的整体网络。
- 权重共享(Weight sharing)：在NAS过程中，最为耗时的其实就是对于候选模型的训练。而初版的NAS因为对每个候选模型都是从头训练的，因此会相当耗时。一个直观的想法是有没有办法让训练好的网络尽可能重用。Google的论文《Efficient Neural Architecture Search via Parameter Sharing》提出了ENAS，其核心思想是让搜索中所有的子模型重用权重。它将NAS的过程看作是在一张大图中找子图，图中的边代表算子操作。基本方法和《Neural Architecture Search with Reinforcement Learning》中的类似，使用基于LSTM的控制器产生候选网络结构，只是这里是决定大图中的哪些边激活，以及使用什么样的操作。这个LSTM控制器的参数和模型参数交替优化。由于权重共享，使用Nvidia GTX 1080Ti可以在一天内完成搜索，实现了1000x的提速。Auto-Keras就是基于ENAS的思想加以改造实现的。最近，中科院的论文《You Only Search Once: Single Shot Neural Architecture Search via Direct Sparse Optimization》中提出了DSO-NAS方法，如其名称其特点是只搜一次。它始于一个完全连接的块，然后在操作间引入缩放因子，同时添加稀疏正则化来去除无用的连接，也就是去除不重要的操作，得到最优结构。文中提出可以在一个优化问题中同时学习网络结构和参数。
- 表现预测(Performance prediction)：我们知道，NAS中最费时的是候选模型的训练，而训练的目的是为了评估该结构的精度。为了得到某个网络模型的精度又不花费太多时间训练，通常会找一些代理测度作为估计量。比如在少量数据集上、或是低分辨率上训练的模型精度，或是训练少量epoch后的模型精度。尽管这会普遍低估精度，但我们要的其实不是其绝对精度估计，而是不同网络间的相对值。换言之，只要能体现不同网络间的优劣关系，是不是绝对精准没关系。论文《Progressive Neural Architecture Search》中的关键点之一是使用了一个代理模型来指导网络结构的搜索。这个代理模型是个LSTM模型，输入为网络结构的变长字符串描述，输出预测的验证精度。还有一个思路就是基于学习曲线来预测，这基于一个直观认识，就是我们在训练时基本训练一段时间看各种指标曲线就能大体判断这个模型是否靠谱。学习曲线预测，一种直观的想法就是外插值，如论文《Speeding up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves》中所讨论的。之后论文《Learning Curve Prediction with Bayesian Neural Networks》中使用贝叶斯神经网络对学习曲线进行建模与预测。当然实际的预测中，可以不局限于单一的特征，如在论文《Accelerating Neural Architecture Search using Performance Prediction》中，结合了网络结构信息，超参数信息和时序上的验证精度信息来进行预测，从而提高预测准确性。总得来说，表现预测由于其高效性，在NAS中起到越来越关键的作用。很多的前沿方法也采用这类方法，如前面提到的《Neural Architecture Optimization》中就有预测器来基于网络表示预测其准确率。

2.2 变体及扩展

上面主要针对NAS本身的改进，且主要实验限于图像分类任务和自然语言处理，且大多只关注模型准确率。很自然的，可以将NAS往其它的任务上应用，或者将更多因素纳入目标考虑。比如下面两项工作：

- 语义分割：Google论文《Searching for Efficient Multi-Scale Architectures for Dense Image Prediction》将NAS拓展到主义分割领域（源码链接）。语义分割任务与图片分类相比，输入的尺寸更大。经典的语义分割网络需要特殊的结构，如空洞卷积、ASPP等来提取多尺度上下文信息。这给NAS带来了更大的挑战。这篇文章主要focus在两点：搜索空间的设计与代理任务的设计。对于搜索空间，文中提出了适合语义分割任务的DPC(Dense Prediction Cell)。对于评估精度的代理任务，由于语义分割任务与图片分类不同，往往依赖大分辨率，因此，代理任务的设计上，一方面采用小的网络backbone；另一方面将在训练集上得到的feature map cache起来重用。实验表明，机器搜到的网络在如Cityscapes、PASCAL-Person-Part和PASCAL VOC2012上能得到SOTA结果。
- 多目标：随着各种AI场景在手机等端设备上应用需求的涌现，一些适用于资源受限环境的轻量级网络，如MobileNet，ShuffleNet开始出现并被广泛研究。自然地，NAS也开始从只考虑精度的单目标演进到多目标，如同时考虑精度、计算量、功耗等。多任务优化问题的一个挑战是往往没法找到单一解让所有子目标同时最优，所以我们找的一般是帕累托最优解。而具体到实际应用中，就涉及到多目标间tradeoff的问题。Google的论文《MnasNet: Platform-Aware Neural Architecture Search for Mobile》提出的MnasNet就是比较典型的例子。搜索方法上还是延续之前经典的RNN控制器+强化学习方法，在一个层次化搜索空间中搜索。它一方面提出带参数的优化目标，通过调节参数可以在精度和延时之间做权衡（如将延时作为硬要求或者软要求）；另一方面在实现中将实际设备运行得到的推理延迟作为评估。从实验来看，能比前沿的轻量级模型找到更优的网络结构（如在ImageNet分类任务中，差不多精度下比MobileNet v2快50%）。

NAS已在一些基本任务及数据集上证明能超越人类手工设计，在其它深度学习相关的超参数上也能使用类似的思想让机器自动学习，从而得到比人工设计更好的参数。其他应用还有如模型压缩、优化器调参、数据增强等。

reference:

http://xcx1024.com/ArtInfo/539466.html

https://zhuanlan.zhihu.com/p/45133026

https://www.tensorinfinity.com/paper_136.html