自上而下思考实现AGI技术难点及可能方法

实现通用人工智能（AGI）道路之我见 I

前言

我对实现通用人工智能（AGI）有着非常浓厚的兴趣，平时经常思考一些实现AGI的最短路径，工作中也尽量选择一些能为AGI长期研究有所帮助的方向。但我深知自己能力有限，很多方面缺乏系统的理论指导。这次我把我的思考写下来，希望能够和大家一起讨论这个议题，得到各位专家的指正。

这一系列分为三篇文章：“自上而下思考实现AGI技术难点及可能方法”，“稀疏是通往AGI的必由之路”，和“AGI，从我做起”。

这篇文章尝试用自上而下的方法探讨实现AGI的技术难点。其中很多思考源自我2017年写的两篇文章：“A few predictions on artificial intelligence”[1]，和“some scribble of things”[2]。因为是自上而下，很多观点源于一些猜测和推断。通俗一点的话就是有点虚。我也希望通过讨论能够变得更实一点。

我的思维方式

我尝试自上而下，从三万米高空俯视，分解智能生物所需的一些能力，以及自下而上的从现今人工智能的现状推测今后发展的路径，从中找出一些结合的技术难点以及可能解决的方向。

我自下而上的思考方式是从现在传统神经网络研究出发，而不是从内脑计算（neuromorphics）出发，但从下面一篇讨论稀疏性的文章来看，这两者的区别不是想象中的那么大。

什么是智能生物

在介绍AGI之前，先要定义AGI。不同人眼里AGI的标准有很大差别。对于我来说，我认为人工智能是一种智能生物，而AGI是人类创造出来的，但又摆脱对其依赖的智能生物。AGI和其他智能生物一样，有一个目的，这个目的就是生存”。而实现这一目的的方法是：

Approximate the environment, the past, the present, and the future.

这中间也有一个矛盾：智能生物是在环境中的，所以智能生物的复杂度要比环境的复杂度小，这么做的难点是：

A lower capacity subject approximates a higher capacity subject

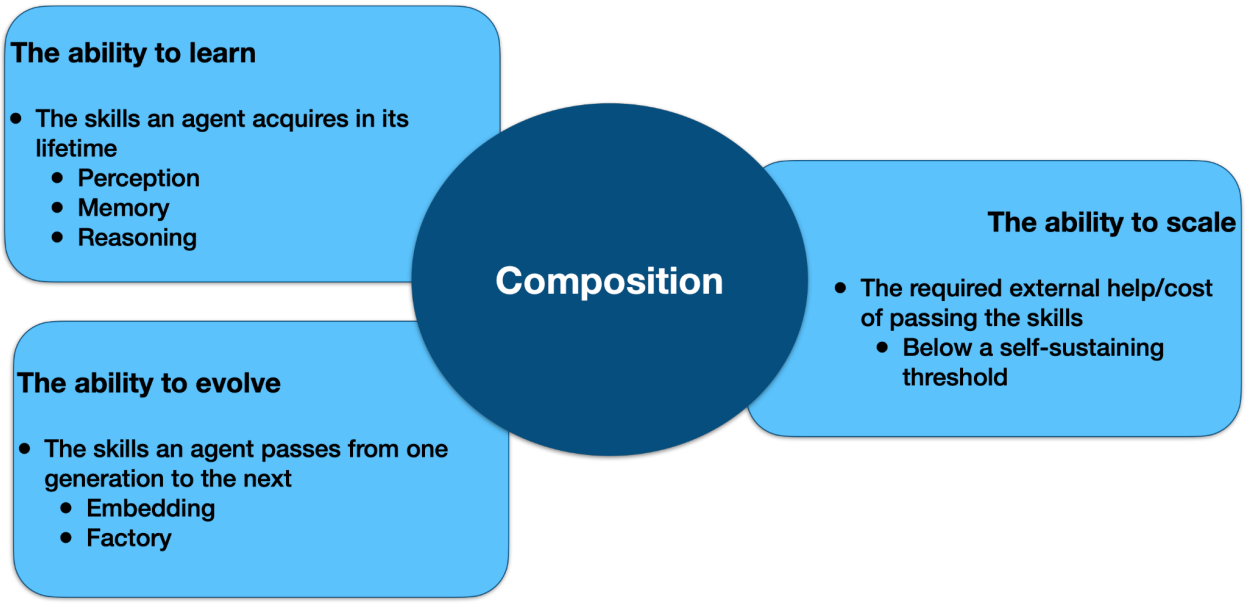
所以智能生物需要对环境做某些近似、建模，从而能够尽可能approximate环境，AGI也不例外。

我们在研究如何实现AGI时主要关注AGI如何更高效的approximate环境，其中可以从人如何高效approximate环境得到一些启示。

智能生物所需的三大能力

一种智能生物，如果要能够在环境中独立生存，需要具有三种能力：学习的能力、进化的能力、和拓展的能力。这三种能力发展的方向和阶段不尽相同，这也是我们世界中有非常多的动植物物种的原因。因为我们探讨的是高度智能的AGI，我们就用人作为样板来论述这三种能力。

如图一所示，学习的能力是一个智能生物个体在之一生中能够获得的技能；进化的能力是智能生物个体将技能传递给后代的能力；拓展的能力是智能生物体能够独立传递技能的能力。在这之中composition起了非常重要的作用。



图一：智能生物所需的三大能力

学习的能力

学习的能力是一个智能生物个体在之一生中能够获得的技能。因为有个体与个体的区别，这里有个假设是个体的结构基本上是固定的。这也就直接导致了个体的结构决定了学习能力。举例来说，AlexNet的效果不如ResNet，就是由不同的结构决定的。

猜想:学习是件容易的事，可以用基于*SGD*的方式获得。

学习有三重递进的组成能力：perception，memory，和reasoning。

Perception

Perception是感知环境的能力。这也是其他学习能力的基础。地球上的生物花了数亿年的时间进化出比较完善的感知能力。而人类大脑的大部分都是在处理感官信息[3]。

我的理解是，perception将环境中实时输入的非常高维的信息做了坐标系变换，将独立的信息转换到垂直的坐标中，从而最终达到降维的目的。这就是DNN中的feature extraction。现在DNN研究中有很大一部分是寻找模型结构以更高效的实现feature extraction。

Memory

我认为memory就是一个时间上的坐标系变换，将perception产生的feature保存下来，并在将来需要的时候将之选择性的读取出来。这是perception的自然延伸。

现在在这个领域并没有很深的研究，大多数模型将学习feature的方法保存在模型的权重中。处理spatial输入时用相同的权重处理一遍（如MLP，CNN）。在处理时序序列时将短期记忆embed在激励中，仍然用相同的权重处理一遍（如RNN）。这样历史学习的feature并没有独立的存在模型中。DeepMind有过一些尝试，如Neural Turing Machine[4]，Differential Neural Computer[5]，将历史feature单独存取，但近期没有很多突破性进展。

长期memory的研究是continual learning的前提，也是实现AGI的必经之路。

Reasoning

Reasoning看上去很神秘，但我认为这只是perception和memory的自然延伸：

> *Reasoning is the process that correlates the spatial features (perception) and the temporal features (memory), and **predicts** future features.*

现在大多数DNN所做的事就是unconscious归纳推理（inductive reasoning）：在训练中看了一千次羊，推理中就能够认出羊了。其中DNN所做的就是perception（认出羊），prediction在训练和推理的时序中体现出来。

当然更高层次的演绎推理（deductive reasoning）就不是这么容易了，光是大前提本身就是prediction。

我倒是觉的reasoning本身并不困难，功夫到了就自然可以学到了。如果和人类类比的话，从最开始有reasoning到现在有这么多的哲学思想仅仅花了几千年。而之前发展出memory和perception花了万亿年。是不是这意味着我们先要着重将perception和memory的发展体系研究透彻呢？

进化的能力

既然学习中个体的结构基本不变，那个体学习的能力是有上限的。如果要持续提高个体的学习能力，就需要显著改变个体的结构。这个过程就是进化。

> *Evolution is the process that significantly changes agents' structures to increase their learning capabilities*

进化是基于学习之上的。如果要编一个提高个体学习能力的程序的话，这是一个双重循环，学习在内循环，而进化在外循环。这也意味着进化比学习要难很多，慢很多，需要多非常多的计算。

进化是否可以用SGD得到呢？

这里我们用SGD来泛指一系列通过一阶导数逼近结果的方法，如SGD with momentum，Adam。我们可以想象学习是在一个高维学习空间用SGD优化得到最优解的过程。同样，学习加进化也可以想象为在一个包括学习和进化的高维空间，那SGD是否能够得到最优解呢？

但这有一个问题：现今进化的方法是离散的，所以不能表示成连续的高维空间。但这不是绝对的，我们可以假想进化的改变也可以是无限小可微分的，这样SGD也是可以适用了。但是，就算是这样，进化所需的外力（cost）和学习的外力是不成比例的（进化需要改变结构），这导致了这个高维空间是山峦起伏的，山谷基本上都是相同结构学习的最优解。Saddle point在这种空间里可能并不常见。从而导致SGD不能解决进化的问题。（[2]对这方面有从另一角度的阐述）

所以我不认为进化可以用SGD来解决。也因为进化的复杂性和无序性，我倾向进化需要非常众多的个体尝试不同方向，然后由自然选择来完成结构的优化。

现阶段进化方法

现阶段绝大多数模型都是人们用自己的聪明才智设计出来的（如ResNet，EfficientNet，Transformer），所以模型只是用来学习，而模型的进化通过一个超过模型能力很多倍的super power（人）来实现。这种进化全部依赖外力实现。

现在人们也在研究neural architecture search（NAS），让程序来进化模型。NASNet[6]用强化学习的方法，AmoebaNet[7]用进化算法的方法来寻找更好的模型结构，这些方法都是普世的寻找结构的方法，但需要消耗非常多的运算（学习在内循环、进化在外循环）。

DARTS[8]，FBNet[9]，将模型搜索转化成模型学习，先设计一个super net的结构，在给定一个限制条件的情况下用GumbelSoftmax来sample合适的模块。Once-for-all[10]又将之发展一大步，训练完super net之后可以得到一连串不同的模型以适应不同的限制条件。你可能会问，这不是用SGD的方法来解决进化问题吗？其实不竟然，主要问题在这个super net，在super net确定的情况下可以用简单的方法来选择子网络，但这并没有解决super net是怎么发现的问题（还是通过外力，人来发现的）。这说明从一个更优解到在一些条件下的次优解是有简单路径的，但并没有解决如何找到这个更优解。

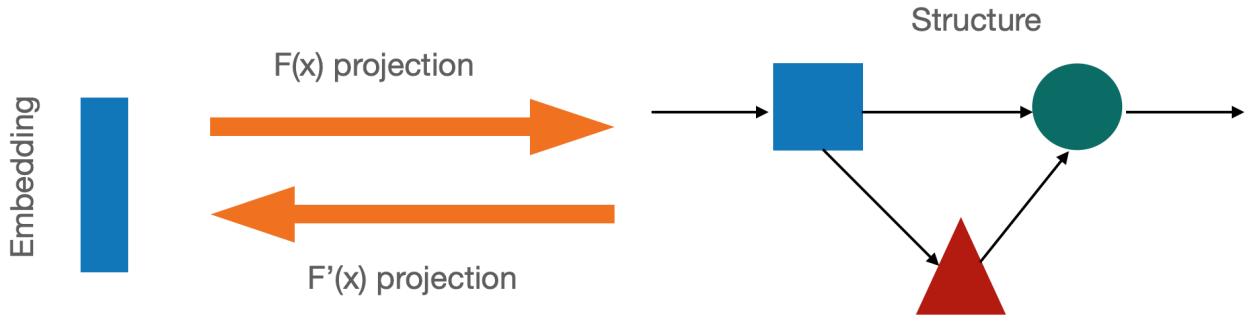
ZenNAS[11]从另一个角度阐述这个问题，用进化算法在不训练单个模型的情况下搜索网络结构，这里搜索还是非常复杂（非SGD），但用了ZenScore来替代训练得到的精度来减少内循环的时间。对特定模型结构显示出很好的效果。

现阶段NAS有两大问题：1，直接搜寻模型结构，但是非常细粒度的模型结构的变量太多了（例如，每一个权重和每一个激励的关系，这是infeasible的）。而现在模型的macro-architecture又是相对简单，就开始寻找模型架构的proxy，这又导致了第二个问题。2，模型搜索的building blocks是人指定的，搜索的规则也是人指定的。

我这里也臆测一下今后模型进化的方式。

Embedding

未来的模型结构如果变得非常复杂，直接对之进行各种改变并记录改变结果是不现实的，因为结构中的自由变量太多了。这时需要用一种方式降低自由变量数，但又不显著改变模型的表达能力，这就是embedding。从embedding到模型结构有一个一一对应的关系，中间需要一个固定的函数做转换。这个固定函数F(x)可以将少数自由变量映射到复杂的结构中去。如图二所示。



图二：由embedding project模型结构

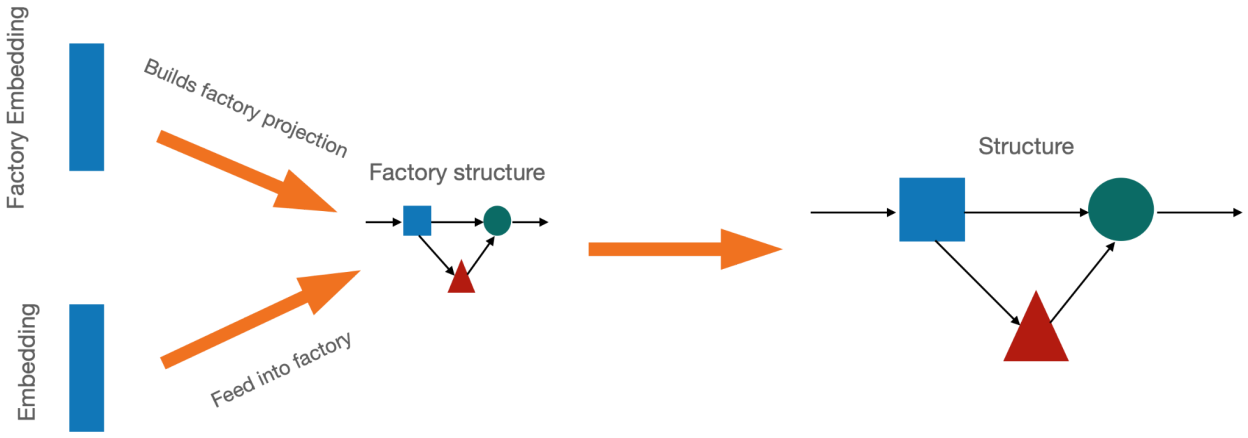
有了embedding之后，就可以用传统的进化算法来做mutation，crossover等操作来进化模型了。

现在NAS已经有了embedding的做法，只是现在模型结构很简单，用几个值作proxy（如宽度，深度等），F(x)就是直接映射，并没有起到减少自由度的作用。

Factory

NAS第二个问题是模型搜索的building blocks是人指定的，搜索的规则也是人指定的。这显著限制了搜索空间。要扩大进化的搜索空间，需要用Factory方法。这里我从"the gang of four"写的著名的design pattern的书[11]里借用了Factory design pattern。Factory design pattern可以生成不同的object，但又不将生成的逻辑暴露给生成的object。

与之类似，要扩大搜索空间，需要embedding和projection函数的联动。因而projection函数就不能是一个固定的函数，而也是可以从embedding生成的。对同一个factory输入不同的embedding的话可以生成不同的结构。如图三所示。



图三：由factory生成模型结构

这样factory的结构也会有变异，从而大大增加最终生成结构的丰富程度。这里产生factory的projection还是一个固定函数，但factory可以是hierarchical的，让这个projection函数也是由第二层的factory结构生成，这样模型的复杂度就可以无限增加了。

拓展的能力

学习和进化在前期都需要外力的协助。就算AI变得非常聪明，但如果外力撤除了，AI还是不能够自己推动自己生存。这样的AI还不能算足够通用（自己不能确保自己的生存，怎么能算通用呢？）

这就需要AI能够自我维持进化，需要更新结构的开销小于更新结构带来的报酬，产生正反馈，并且超过零界点。同时，在进化中能够保持足够的差异性和冗余性，不会因少数核心部分被攻击而全体灭亡。

现在这一能力还没有涉及，可能今后很久都不会涉及。等到AI能够较好的完成学习和进化再考虑这一点吧。

Composition

Composition是一种大规模复制的方法，这是在学习和进化中用少量的开销而显著增加复杂度的有效方法。很不幸的是现在的DNN并不能利用composition，这成为增加模型表达能力的主要缺失。我认为，如果不能找到有效compose的方法，要达到AGI的复杂度是几乎不可能的。下面我介绍一下我认为的composition的方式和难点。

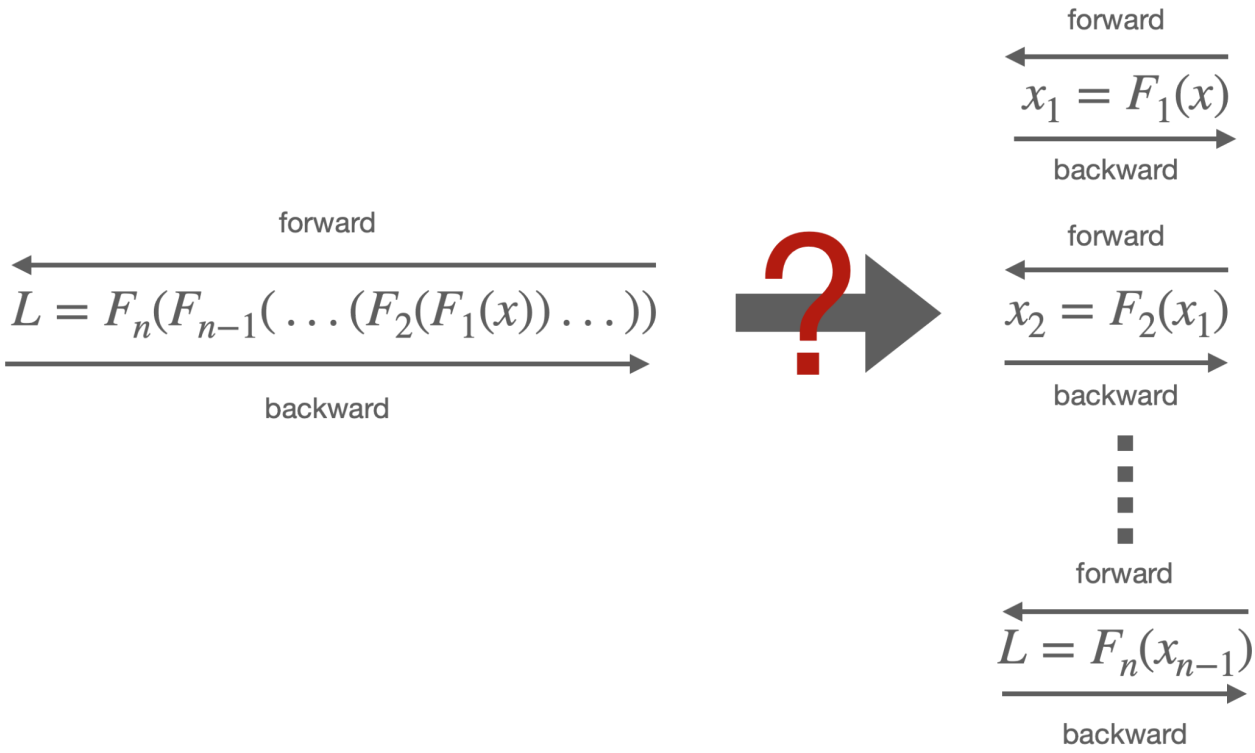
Function composition

一种composition叫做function composition，其核心是一个函数可以表达为先后施加两个函数，如：y = f(g(x))，这样 f 和 g 就可以复用了。DNN模型其实可以表达为：

$$L = F_n(F_{n-1}(...(F_2(F_1(x))...))$$

其中F_i是模型第i层的函数。这本身就是一种function composition。但是，不同于一般的函数，我们可以复用$F_n...F_1$吗？

如图四所示，我们需要decompose模型的训练为每一层的训练，这样才能复用模型的一层在其他模型上。现在的模型中，transfer learning有一点点这么做的影子，像object detection，instance segmentation里的backbone就是从classification里训练出来。以及需要pretrain的模型，如BERT，在encoder后面加上不同的decoder就可以用在不同的应用上。但这种方法很难扩展到将多个部分compose起来。在这方面还有很多工作要做。



图四：Function compose一个DNN模型？

Object composition

Object是一种比function更抽象的表达方式，用来包装数据和内部关系。从某种意义上来说，object的本身意味着object内部的联系比外部的联系更为紧密。我们可以用聚类的方式将表征关系的高维空间里临近的点聚类为同一个object，更为稀疏联系的点分为不同的object。Object可以分层的，compound object就是将关系较为紧密的basic objects包装起来，组成新的object。

Object的目的也是复用，这些object可以用在不同的目的，或者作为同一目的不同个体，而不需要重新从零生成了。

我看现在在ML领域还没有采用object这种方式。

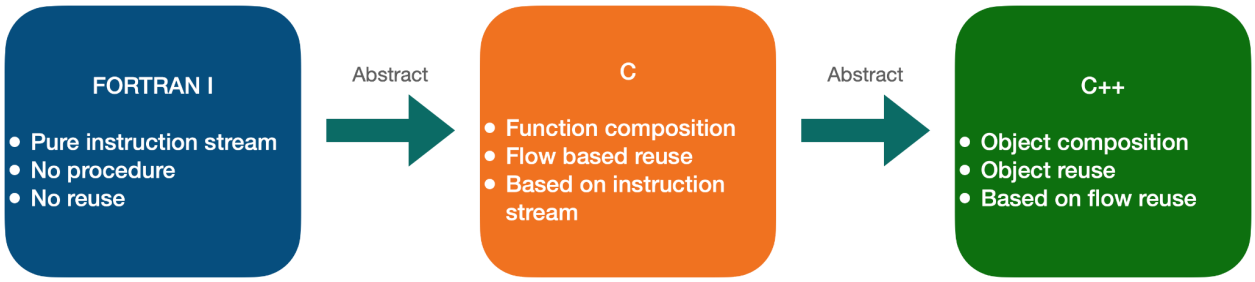
未来模型Composition之路

现在对DNN的处理是monolithic的，将所有的东西混在一起，不加区分。这和composition是完全相反的方法。

这种方法的好处是，在模型大小受限的情况下性能可以很好。这是因为monolithic的方式不需要复用，可以对每一个最小的单元做很细微的调整，从而最大限度的提高性能。这是用composition做不到的，因为composition意味着复用，而复用的代价是牺牲一部分性能。

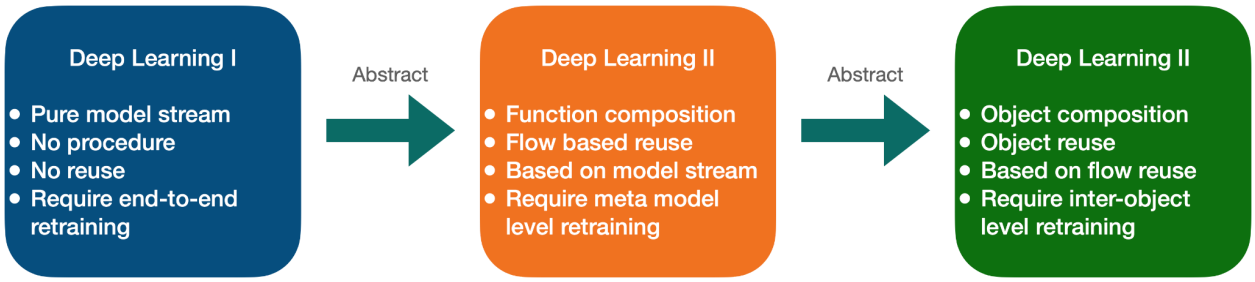
然而这种方式的弊端也很明显，就是模型的大小是受限的。没有复用机制大幅增加模型复杂度是非常困难的。而composition所带来的性能损失可以通过增大模型复杂度来弥补。从软件工程和硬件设计的历史来看，monolithic到一定程度一定需要通过composition的方式来大幅提高复杂度。

从软件工程的方式来说，我们现在广泛使用的软件编程方式是软件1.0：人手工指挥机器一步一步进行计算。这中间也经历了function composition和object composition的阶段，其中C语言是一种function composition的语言，而C++是object composition的语言。如图五所示。



图五：软件1.0的抽象过程

现在有一种说法将DNN模型看作软件2.0 [12]：人们不再纠结于一步一步的指令了，而是负责提供数据，结果就出来了。中间的过程就由DNN模型来承担。在这种情形下我们可以和软件1.0作一个类比，看看我们在哪一个阶段。如图六所示。



图六：软件2.0的抽象过程

我们现在还在最原始的第一阶段，而且我们还在不择手段的用monolithic的方法提高模型复杂度，是不是我们需要后退一步，思考一下用composition的方法能不能得到比现在大很多的模型呢？

同样，我们也可以回想一下硬件设计的历史。硬件设计是从模拟电路开始，然后发展到数字电路，然后就一发而不可收拾。在图七，我们将模拟电路、数字电路和现在的深度学习比较一下。

	Deep learning	Analog circuit	Digital circuit
Foundation	Back propagation and chain rule	Semiconductor physics	Boolean algebra
Impact of changes local or global?	Global	Global	Local
Can be composed?	No	No	Yes
Abstraction hierarchy	No	No	Device, circuit, gate, RTL, IP, SoC
Scalable?	No	No	Yes

图七：深度学习和模拟、数字电路的比较

从中我们可以很明显的发现，现在的深度学习和模拟电路非常类似，而和数字电路有很大的差别。虽然模拟电路在过去的一个世纪中取得了长足进步，但真正翻天覆地的改变世界的是数字电路。

对比深度学习和数字电路，我们现在还缺少实现composition的理论基础。类似于boolean algebra对数字电路提供的理论基础，我们需要找到针对深度学习的理论基础。请注意这不是探寻深度学习为什么能工作。探寻深度学习工作原理就像寻找模拟电路的理论支持一样（半导体物理）。

从另一方面来讲，模拟电路是数字电路的基础。数字电路将模拟电路产生的数值离散化（0/1），进行抽象，并基于0/1搭建出来的一套理论。但模拟和数字电路是同源的。同一个MOSFET，如果我们使用它的线性工作区间，这就是模拟电路，如果我们使用它的饱和区间，就变成了数字电路。为了保证数字电路都在饱和区工作，我们在EDA中设计了众多的design rule。但是，在深度学习里我们能够找到相似的特性吗？

总结

虽然现在深度学习已经在非常多领域里崭露头角，但从一个更长远的视角来说，深度学习的起飞才刚刚开始，我们甚至还没有掌握起飞的敲门砖：composition。

Composition的核心是复用。只有通过composition，我们才能设计出人脑级别复杂的模型；只有通过composition，我们才能通过众多的个体实现结构的进化；只有通过composition，我们才能通过拓展，迅速降低结构演变的开销。

我们是不是需要研究一下composition在深度学习中的存在方式呢？

References

[1] https://feisun.org/2017/12/24/a-few-predictions-on-artificial-intelligence/
[2] https://feisun.org/2017/12/24/some-scribble-of-things/
[3] https://en.wikipedia.org/wiki/Perception
[4] https://arxiv.org/pdf/1410.5401.pdf
[5] https://deepmind.com/blog/article/differentiable-neural-computers
[6] https://arxiv.org/pdf/1707.07012.pdf
[7] https://arxiv.org/pdf/1802.01548.pdf
[8] https://arxiv.org/pdf/1806.09055.pdf
[9] https://arxiv.org/pdf/1812.03443.pdf
[10] https://arxiv.org/pdf/1908.09791.pdf
[11] https://springframework.guru/gang-of-four-design-patterns/
[12] https://hazyresearch.stanford.edu/software2