**在深层强化学习中，状态空间分解和子目标创建**

**摘要**

典型的强化学习(RL)代理学习完成根据他们的领域指定的奖励功能指定的任务。因此，他们学到的政策并不能推广到类似的领域。为了解决这个问题，我们开发了一个框架，在这个框架中，一个深度RL代理可以使用一个重复的注意机制，从更小的、更简单的域，到更复杂的域。任务以图像和指定目标的指令呈现给代理。这个元控制器通过在状态空间中设计一个较小的子任务序列来引导代理实现它的目标，从而有效地分解它。作为一个基准，我们考虑的是一个没有注意的设置。我们的实验表明，元控制器学会在关注的范围内创建子目标。

**关键词**:等级强化学习，转移学习，政策梯度，注意机制

**State Space Decomposition and Subgoal Creation for Transfer in Deep Reinforcement Learning**

**Abstract**

Typical reinforcement learning (RL) agents learn to complete tasks specified by reward functions tailored to their domain. As such, the policies they learn do not generalize even to similar domains. To address this issue, we develop a framework through which a deep RL agent learns to generalize policies from smaller, simpler domains to more complex ones using a recurrent attention mechanism. The task is presented to the agent as an image and an instruction specifying the goal. This meta-controller guides the agent towards its goal by designing a sequence of smaller subtasks on the part of the state space within the attention, effectively decomposing it. As a baseline, we consider a setup without attention as well. Our experiments show that the meta-controller learns to create subgoals within the attention.

arXiv:1705.08997v1 [cs.AI] 24 May 2017

**Keywords:** Hierarchical Reinforcement Learning, Transfer Learning, Policy Gradient, Attention Mechanism

# 介绍

通常，强化学习(RL)代理不能将小领域学到的策略推广到更大、更复杂的领域。然而，这种能力可以让他们立即应用在简单设置中学习的技能，而不必去探索一个大的州空间。在深度学习环境中，减少国家的规模也意味着需要更小的网络，这更容易进行培训。在这个工作中，我们提出了一种方法，将复杂的环境分解为更简单的环境，并在其中提供子目标，从而最终解决更大的任务。代理可以在较小的环境中预先训练，独立地解决每个子目标，或者与子目标创建算法相结合。

我们描述了一个元控制器，它学习分解状态空间，并在较小的空间内提供可解析的子目标。元控制器正在解决一个延迟的奖赏问题，因为当底层代理解决原始任务时，它只得到正强化。它必须提出一系列的子目标，从而使这种强化的期望最大化。除了创建子目标之外，元控制器还会对状态空间进行碎片化，使底层代理呈现一个较小的状态，从而可以轻松地为子目标学习一个最优策略。它通过使用一种注意力机制来完成这一过程，类似于重复的注意力模式1。元控制器学会控制它的注意力，并且只将状态的一部分传递给代理。元控制器的MDP公式是:

* 状态, *S*,是过去和现在的关注的总结.
* 行动, *A*, 是Lattn注意力的位置*Lattn*, 以及一套子目标的分配, **g**.
* 奖励, *r*, 如果底层的代理能够解决这个任务并且一个小的负的步骤成本，那么奖励r是正的
* 转换:底层代理根据提供的状态和子目标执行其策略。由于这个策略对元控制来说是未知的，所以这是它环境中的一种可施性的来源.

元控制器为Lattn和分配P(g)选择一个值。在那个位置下的状态空间和一个子目标g，被传递给底层代理。然后，代理选择一个将其移动到实现g的原子动作。新的代理位置Lagent改变了元控制器的环境，它选择了一个新的关注点和子目标。

在这项工作中，我们做了一些简化的假设。首先，我们假设底层代理可以访问每个子目标的最优策略。这种依赖于目标的策略可以通过诸如通用价值函数近似(UVFAs)2这样的技术来学习。UVFAs学习接近V(S，g)或关于目标的值函数，使用类似于深度神经网络的函数近似。学习价值函数V(S，g)可以用来构造一个实现目标g的策略。这个值函数可以独立于或与元相结合，为实现子目标3提供内在的奖励。其次，我们假设代理仍然保持状态，除非它的位置和子目标都是由元控制器提供给它的状态。一般来说，元控制器会自动被激励去集中注意力，并提供子目标，这样潜在的代理就能够解决给定的任务，因为它的奖励结构。在这种情况下，这意味着在注意中保持代理位置和子目标。例如，在Pacman的游戏中，如果子目标是吃最接近的药丸，那么潜在的代理应该有Pacman和至少一种药丸。否则，代理可能会随机移动，它将无法实现获得高分的总体目标。

上面的假设简化了元控制器的训练，但是我们提供的方法应该适用于基本的设置，也就是底层代理的策略也得到了学习。.

# 相关工作

我们的工作与Kulkarni等人的工作最接近。它们提供了一个层次结构框架，在这个框架中，代理可以从更高级别的代理设置子目标并在较长时间框架内运行的内在奖励中学习。高级代理的奖励是由环境提供完成任务所提供的。子目标依次通过在面向对象框架中的实体和关系来提供。从某种意义上说，我们的方法进一步分解了状态空间，使基本代理只能在任何时候看到它的一小部分。这允许更好的计算效率，因为基础代理现在可以使用更小的网络，并且可以允许将已学习的策略转移到状态空间的不同部分，而不必显式地探索它们。为了实现这一点，更高级别的代理，或者元控制器，必须学会将信息集成到目前为止所观察到的状态。因此，我们使用一个循环模型来通过长短期内存(LSTM)网络4来表示元控制器。Kulkarni等人使用了一对dqn作为代理和元控制器。

为了训练我们的元控制器的注意力机制，我们采用了一种类似于Mnih等人1的技术。他们使用策略梯度来训练分类和简单控制任务的注意力机制。在我们的方法中，我们不使用复杂的瞥一眼传感器，而是简单地使用5x5的输入图像。这可以很容易地合并到我们的设置中。此外，我们没有直接使用连续的输出，而是使用离散的动作，向上，向下，noop，以转移注意力。

最后，Schaul等人描述了如何为深的RL代理构建目标特定的函数。这样的一个函数可以在5x5图像上独立学习子目标，从而为我们的基础学习者构建。我们没有为本文的设置提供结果，但可以将其集成到未来的工作中。.

# 初期

B在继续进行之前，我们将简要介绍强化学习和政策梯度。

## 强化学习和MDPs

强化学习解决了选择行为的问题，这种行为最大化了长期累积奖励的概念。它通常被制定为Markov的决策过程(MDP)。MDP的特征是元组小于S，A，T，R。S是代理可以进入的状态集。A(S)是代理在每个状态中都可以使用的操作集。通常，代理选择一个从A执行的操作，根据转换函数T:S(S)，它可以将其带入一个新的状态。R:S A R是在一个状态下执行一个动作时接收到的标量值。最后，0 1是折现率。策略，即S P(A)，通知代理在每个状态中执行哪个操作。强化学习代理的目标是找到最优策略，使每个州的长期预期回报或效用最大化。

## 梯度策略

梯度策略方法通过调整政策参数直接直接最大化预期回报。预期的奖励一个轨迹采样从政策πθ是由参数化

*T*

*J*(*θ*) = E*p*(*S*1:*T*;*θ*) X*rt* = E*p*(*S*1:*T*;*θ*)[*R*] (1)

*t*=1

当p(S1:T;θ)取决于国家分布由πR的返回轨道。在这个公式中，没有进行折现。策略梯度定理[5]给出了J的梯度

*T*

∇*θJ*(*θ*) = E*p*(*S*1:*T*;*θ*) X∇*θlog*[*π*(*at*|*s*1:*t*;*θ*)]*Rt* (2)

*t*=1

我们可以从当前政策中取样一组轨迹，并对其进行平均梯度。这是强化算法[6]。与蒙特卡罗样品相比，香草的政策梯度可能会有很大的差异。减少方差的一个常见的修改是从返回中减去基线。基线bt的计算方法是在过去的N个时间步骤中采取观察到的平均回报:。在这项工作中，我们选择N = 100。

# 环境

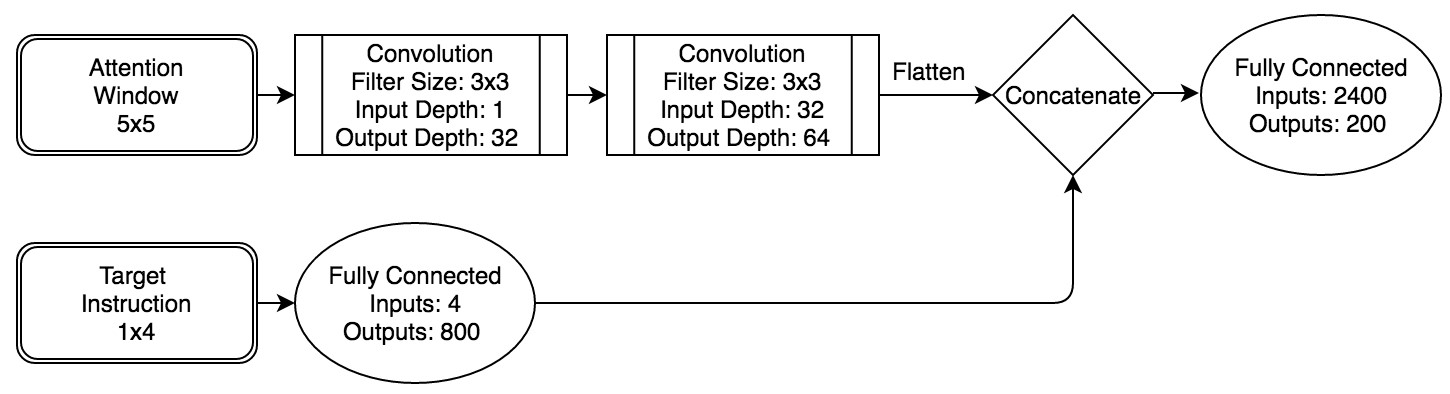
对于我们的实验，我们使用一个包含10x5网格的环境。网格由四个“房间”组成，每个房间都是一个水平的kx5条，一些k不超过4。这些房间层层叠叠，每一个房间都有不同的颜色，分别是红色、绿色、蓝色和黄色。环境还生成一个指令，作为长度为4的一个热向量，指定目标空间。当代理到达目标房间时，一个事件终止，收到+ 1的正报酬，或者它超时而不到达。步骤成本−0.01。

# 方法

我们为元控制器构造三个框架，负责向底层代理提供子目标，这样它就能成功地导航到目标房间。在所有的实验中,亚当meta-controller使用优化器的学习速率1×e−5。代理和注意(如果使用)总是在网格的左上角开始。

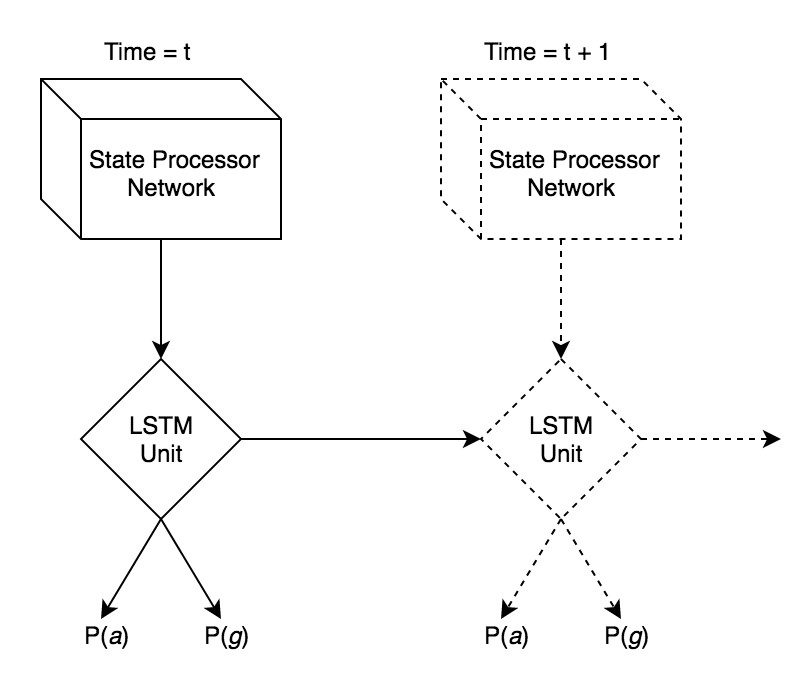
## 元控制器中没有注意机制

首先，我们将整个状态空间作为每个阶段的元控制器的输入来简化问题，因此不使用任何注意机制。具体来说，元控制器接收作为输入的10x5的网格图像，其中包含房间和代理的位置，Lagent。元控制器的输出是一个分布在房间的P(g)。从P(g)中抽取一个房间，并作为指令提供给基本代理。这是它必须实现的子目标。假定底层代理可以在给定指令的整个10x5格上最优地移动。在这个设置中，元控制器的最优策略是直接输出目标房间。



(

a) 状态处理网络



(b) A注意力的元控制体系

图1:部分状态分解和约束注意机制实验的网络体系结构。5 x5注意窗口和目标指令在每次进入元控制器的过程中被输入，然后输出概率分布，而不是注意力操作和子目标指令。

## 元控制器的部分状态分解

在这个设置中，我们有一个带有注意机制的元控制器，它由一个5x5窗口组成，进入10x5网格。除了子目标指令之外，它还必须输出一个动作来控制注意力。在这里，注意机制只部分地分解状态空间，这意味着代理可能最优地移动到提供的子目标，即使代理不在当前的关注范围内。然而，子目标必须位于注意的内部。元控制器的目标是使用它的注意机制来找到目标房间的位置，然后指示代理在每一步的时候进入房间的颜色。

这个元控制器的体系结构由一个状态处理器网络组成，该网络将环境作为输入在每个时间步骤中提供5x5的注意窗口和目标指令。它使用前馈卷积网络处理这些输入，并使用LSTM单元输出P(g)和P(a)，P(a)是关于注意力动作的概率分布。网络的卷积层使用了纠正的线性单元激活函数。注意操作影响下一个注意位置，Lattn，而子目标指令影响下一个代理位置，Lagent。因此，LSTM的隐藏状态包含了从在一集中获取指令和注意力动作序列中获得的知识。为了有效地利用政策梯度来训练这个网络，我们假设注意和指令操作的概率是相互独立的。.

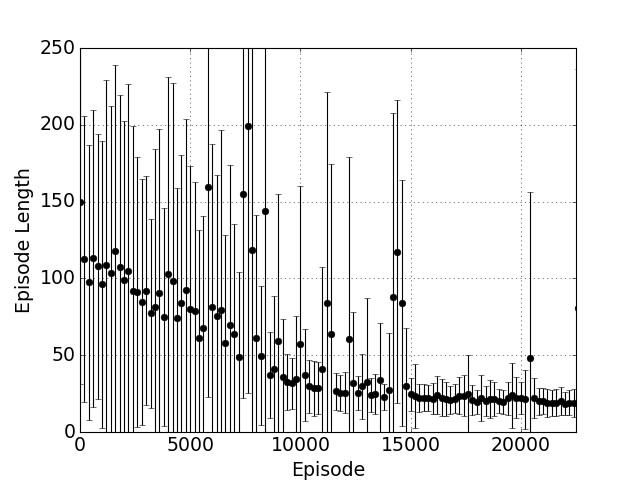
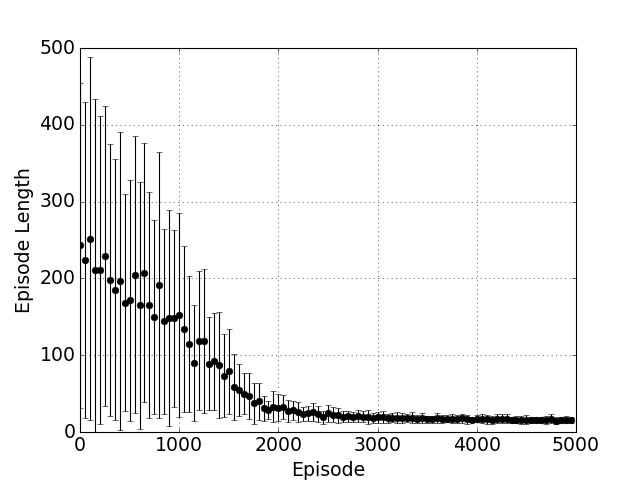
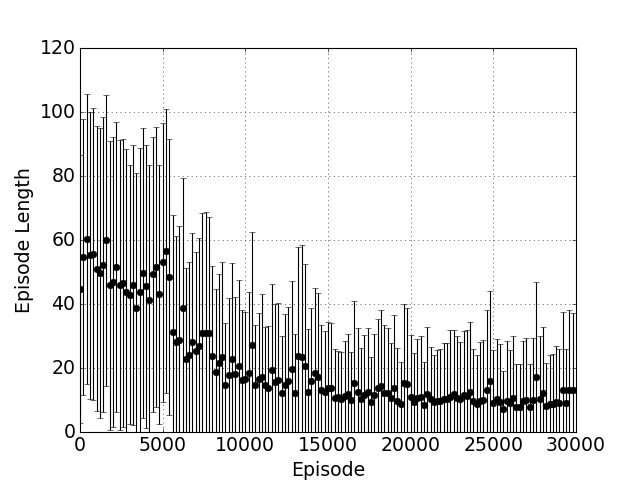
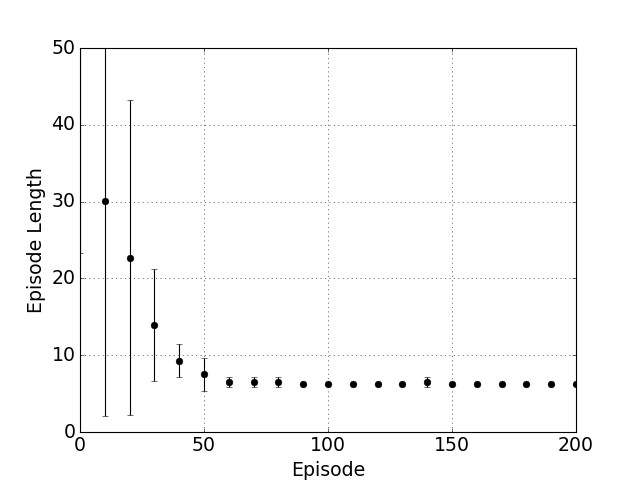
## 元控制器具有有限的注意机制

在这个设置中，除非在注意范围内，否则代理不会移动。这意味着，元控制器必须指示代理在向下移动之前移动到一个房间内，然后重复这个过程，直到代理到达目标房间为止。因此，如果代理或子目标不出现在已分解的状态空间内，则不会实现目标任务。部分状态分解和约束机制实验注意,LSTM单位允许meta-controller使用代理和目标的位置的内存空间来指导其行为选择代理或目标房间内不存在注意窗口在一个特定的时间步。受约束的注意力机械论框架增加了为代理达到目标房间的子目标的最优序列的额外步骤，这是本文的总体目标。

# 结果

## 元控制器没有注意机制

我们用元控制器进行了两个实验，没有注意机制。在第一个实验中，环境是固定的，即房间安排是固定在一段之间的，而目标房间总是在底部。元控制器的最优策略是输出与目标房间相对应的指令，因为底层代理是整个网格中最优的代理。在第二个实验中，环境是动态的，这意味着房间安排是在几集之间随机生成的，但是目标房间总是位于最下面。在这里，它必须学习到最底空间的颜色和最优指令之间的映射。这些实验作为我们使用元控制器和注意机制运行的实验的基线。对于这两种情况，元控制器都收敛到最优策略，引导代理到正确的房间。



(a)元控制器在固定环境 (b) 在动态环境中没有 (c) 带有部分状态分解的元控制器 (d) 具有约束注意机制 中没有注意机制 注意机制的元控制器 的元控制器

图2:训练片段的长度。这些点表示的是根据显示的总集数和须表示方差的变量数的变化数。在没有注意(a和b)的元控制器的情况下，它在一个固定的环境中快速收敛，为代理提供目标房间，从而在最短时间内完成事件。在动态情况下收敛需要更长的时间。带有注意情节的元控制器(c和d)显示了使用注意指导子目标创建的效果。对于这些实验，环境是固定不变的。

## 元控制器与注意机制

在此，我们将对元控制器进行部分状态分解和约束注意机制的结果。在这些实验中，我们把环境固定在每一集之间。与图2a相比，在这两种情况下，将元控制器训练成输出子目标，从而导致最优策略需要花费更长的时间。原因之一是，元控制器现在除了创建子目标之外还必须控制它的注意力。它也在部分观察到的环境中运行，并且必须将过去注意的信息整合到它的隐藏状态。注意，元控制器的状态空间是注意窗口的组合、目标指令和隐藏状态的LSTM单元。但由于这是可以学习的，因此底层代理不必对整个10x5图像进行培训，但只需要对5x5的注意力大小进行培训。我们的方法可能扩展到更大的域，在那里直接学习原始输入图像的策略可能是不可行的。

# 结论

我们的总体贡献是一个框架，允许代理在一个大型的环境中完成一个任务，在更小的环境中了解如何这样做。通过使用注意机制，需要更小的网络，更容易进行培训。在开发了这三个框架后，元控制器了解了房间颜色的表示方式，以及如何将表示转移到引导代理达到所需目标的子指令。我们的结果表明，通过使用注意机制来分解一个大的状态空间，可以在较小的环境中对策略进行扩展。我们的最终目标是与元控制器一起训练底层代理，并将此框架应用于动态和复杂的环境。

# 引用

1. Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. *CoRR*, abs/1406.6247, 2014.
2. Tom Schaul, Dan Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML’15, pages 1312– 1320. JMLR.org, 2015.
3. Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *CoRR*, abs/1604.06057, 2016.
4. Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory.¨ *Neural Comput.*, 9(8):1735–1780, November 1997.
5. Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.
6. Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.