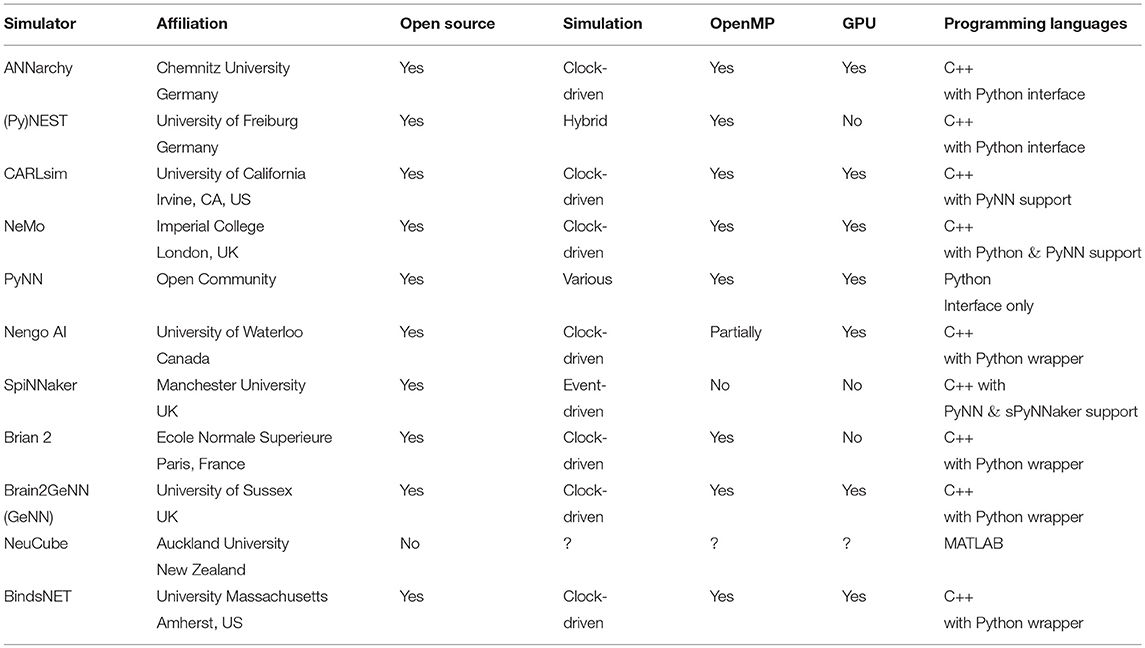
BindsNet是在2018年在Frontiers of Neuroinformation期刊上发表的一篇文章当中所实现的一个用于建立脉冲神经网络的工具包。这个包是基于Python3语言，建立在Pytorch这个著名的深度学习库的基础之上，具有极为友善，语句极为精简的特点，能够快速建立神经网络模型并且进行仿真。而来源于pytorch这一特点也使得这一网络能够在快速的CPU或者GPU单元的快速计算平台上面去进行学习和测试。同时呢，这个工具包在轻微地修改之后也可以适用于不同的软件或者硬件平台，可以在tensorflow上或者在Spinnaker硬件平台去运行。

当前存在许多的SNN仿真的平台，诸如BRIAN2, ANNarchy等等，然而他们普遍存在两个问题。第一点是需要掌握的语句太过复杂，对研究者不友好，第二点是太过关注在生物学上的可行性和真实性，导致模型非常复杂，不适用于需要大规模计算的机器学习和深度学习问题。同时，这些真实的模型也很难大规模在数字电路硬件上面去实现。本工具包的开发者提出了几点对一个仿真网络包所需要具备的特质：易于开发，易于debug，易于运行，同时需要支持用户需要的生物复杂度。同时他们也提出，这种工具应当使用一种恒定开发语言。



以上为一些当前较为知名的SNN仿真平台，PyNEST, BRIAN和ANNarch都是尝试去进行准确的神经元仿真，用于给神经科学研究者构建多层神经元模型，每层神经元都可以指定功能以及生物学特征，同时也可以自己指定连接方式。然而，他们都具有一个限制，就是每个网络当中的神经元类型和突触定义都必须是同质的，如果要去建立一些有多种神经元层，多种连接类型的网络就非常困难。同时，ANNarchy以及BRIAN之类的工具包可以让用户自定义神经元的微分方程，并且通过C++或者Numpy的数值分析方法去进行快速编码，这一点是BindsNet所不具备的，要写新的神经元就需要重新定义一个Node类型。而像Nengo这样的工具包主要是用于仿真高层的神经系统的行动，诸如大脑特定皮层的认知活动等等，同时它也支持一种更复杂更完备的神经元模型，即Izhikevich神经元。GeNN库将网络通过一个C类型的API进行转换，第一次支持了在GPU上实现网络仿真。而以上的大多数工具包大多是利用C++来做底层实现以提供更快的运算速度，而向用户开放的接口则是使用Python或者MATLAB来编写的。而BindsNet的创新之处在于：

1. 依赖Pytorch库，使用其强大的矩阵运算功能来加速SNN仿真
2. 通过创建一个API，可以直接将BindsNet而非其依靠的Pytorch库在FPGA,ASIC,DSP,ARM等等硬件上执行。

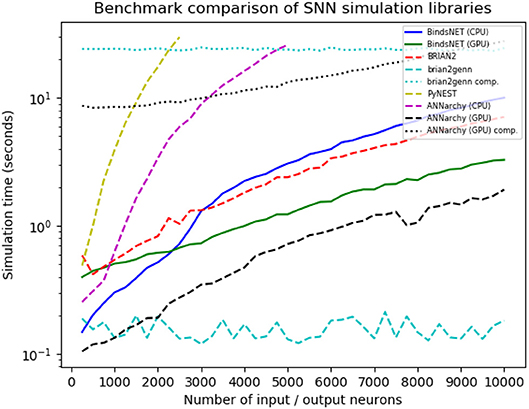
接下来简要介绍这个工具包的文件内容。最重要的网络实现部分在nodes.py和topology.py当中，nodes.py建立单层的神经元，topology.py提供层与层之间的各类连接方式。而最后构造的一个类就是一个network，支持使用monitor当中提供的接口对这个network进行参数监控。Network类型只接受一个参数，就是dt，也就是一个timestep的长度。Network有一个重要成员函数即run(inpts, time) 即对特定时长的输入前向传播time时长，并且进行一个同步时序的权重更新。Reset\_()函数则将该网络的所有参数重置。

Nodes.py当中具体规范了如何建立一层神经元，并且进行单个timestep内的前向传播，以及如何将一层神经元参数置零。当BindsNet内设的神经元包括McCulloch-Pitts神经元，IF与LIF神经元，Izhikevich神经元以及适应性阈值神经元，同时诸多参数诸如阈值，重置参数，静息电位以及不应期等都是可以定义的。

Topology.py当中定义了诸多nodes之间的连接方式，包括普通全连接，2维卷积层之间的而链接以及局部链接等多种链接结构。而这些链接也接受一个update\_rule参数，用于定义其学习方法。现在BindsNet支持的学习方法包括传统的STDP，Hebbian以及常用于强化学习的奖赏调制STDP。而connection当中的norm常数则决定了对单个后突触神经元，所有突触的权重之和，从而起到了权重归一化的作用。

在学习过程中，所有learning.py中所创建的学习类都可以对所有类型的connection使用。而datasets.py调用了pytorch中的模块，用于下载以及预处理应该输入给神经网络的输入。MNIST，spoken-MNIST以及各种CIFAR集都得到支持。而在encoding模块当中，则将这些实数矩阵数据根据其信息强度进行编码，现在支持的编码方式包括泊松编码（速率编码）以及rank\_order()编码，即将时域上的特定时刻与信息强度建立映射。Environment模块主要用于提供强化学习的环境，这一块我还没有调研得特别充分，可以作为下一阶段的任务。而在Evaluation当中，提供了机器学习当中常用的预测以及分类的功能。在无监督学习当中，evaluation模块就提供了用于给予分类的assign\_label()函数以及两种用于预测测试数据的函数all\_activity()以及proportion\_weighting()，前者直接根据输出层当中平均发放脉冲速率最高的神经元来确定预测的分类，而后者会通过现已训练过各类型的数据个数去进行相应的加权来进一步预测。同时，BindsNet也支持进行线性回归与预测的学习评价方法。

而BindsNet提供的视觉化接口包括了一下的内容：画出某次训练中的所有神经元的脉冲发放点以及电压变化色图；画出输入编码后的输入脉冲序列；画出权重的变化色图以及画出性能参数随着学习推进的变化曲线。其他来自matplotlib的方法也可以自行调用。

总的来看，BindsNet由于其开源性以及其基于Pytorch的性质，能够很好、很快地解决运用SNN的机器学习算法在CPU/GPU平台进行仿真的需求，它所提供的预置神经层与连接形式类型以及便捷的编写方式也让之后验证新的SNN结构以及相关算法变得非常便捷。目前BindsNet开发者已经成功仿真了SNN的监督与无监督学习，强化学习，Resoirvoir运算等各种传统机器学习任务，代码可读性也非常高。同时，BindsNet平台的运行速度也是顶尖的，

在GPU平台上在大规模神经网络的仿真的性能方面较之其他平台有较大的优势，在性能和代码书写难度上达到了一个很好的平衡。接下来待调研的问题包括：

1. 调研其在进行强化学习方面的相关模块，包括pipeline.py等
2. 想办法写出一个用于直接将BindsNet移植到其他硬件平台的API。