​**如何在MATLAB中调用已经在python中训练好的深度学习模型**

对于在MATLAB中使用已经基于pytorch训练好的深度学习模型，首先基于标准的网络架构可直接使用MATLAB官方的命令（**参见方案一**）；对于适应不同任务所设计的异构深度学习模型，有一些算子在MATLAB中底层中还没有优化，需要联合python去使用深度学习模型，这样也避免了模型在MATLAB无法初始化等等问题（**参见方案二**）

**方案一**

使用MATLAB官方推荐的deep learning toolbox，其中有详细的示例

<https://www.mathworks.com/help/deeplearning/examples.html?s_tid=CRUX_topnav&category=deep-learning-with-simulink>

或者可以参考如下我的代码

（1）首先需要将训练完成模型文件（.pt文件）转为MATLAB可支持识别的模型文件，这里称为Trace包装器以及脚本化版本（脚本化关键代码：scripted\_model = torch.jit.script(model)）。

可参考如附件代码ptToMatlab\_TorchScript.py，将深度学习模型转为MATLAB可支持的。

（2）再直接导入MATLAB中使用，如下是MATLAB的测试导入使用的代码（我这里的深度学习模型是一个**回归预测深度学习模型**），如下代码可以测试上述转换过后的模型是否可以导入MATLAB使用，我这里给了一个测试输入（test\_input\_raw ），看看能不能导入以及是否可以使用深度学习模型进行预测，完整代码参见NNmodel.m



**方案二**

为适应不同任务所设计的异构深度学习模型，避免出现底层算子不兼容等问题，我们可以使用如下代码在MATLAB中调用python下的深度学习模型。我们依然使用的上述回归深度学习模型

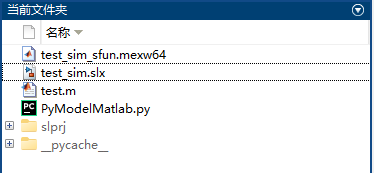
（1）由于训练深度时，我们通常需要配置pytorch、cuda等安装包，再加上Anaconda一起配合使用，从而训练出我们所需的深度学习模型。那么MATLAB调用python下的深度学习模型也需要将MATLAB的训练深度学习模型一样的python环境：

pyenv('version', 'C:\Anaconda3\envs\myenv\python.exe')



上述代码可将默认 Python 环境设置为指定路径的 Python 解释器。

（2）将上述在python中使用回归预测深度学习模型的代码封装成一个函数（def predict(test\_data)：），我这里将这个python中使用深度学习模型的代码命名为PyModelMatlab.py，并将测试MATLAB代码文件命名为test.m，两者都放在统一路径下：

​

（3）在MATLAB中调用回归预测模型代码，即test.m的完整代码如下：

setenv('KMP\_DUPLICATE\_LIB\_OK', 'TRUE');%解决动态链接库（DLL）冲突问题，解决 MKL 库冲突

u = [4 0.016 1024 2.294 0.5 1.15 0.48 0.49 5.00E+10 20 2.5 3400 2900 1900 350];%input tets data

predict\_module = py.importlib.import\_module('PyModelMatlab');

py.importlib.reload(predict\_module);

% 显式转换为NumPy数组并指定数据类型

u\_np = py.numpy.array(u, 'float64');%转化匹配python的数据格式

y = double(py.PyModelMatlab.predict(u\_np));%将预测结果转为double类型

disp(double(y))



（4）simulink中使用上述模型

        在simulink使用python下的深度学习模型与直接在MATLAB命令行使用是有一些不同的，需要为导入python库的代码提前在simulink中进行声明，声明外部函数，告诉代码生成器不需要为其生成C代码：

coder.extrinsic('py.importlib.import\_module');

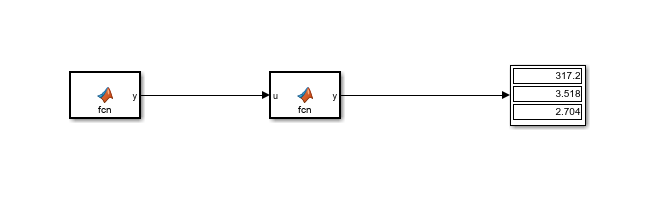
coder.extrinsic('py.importlib.reload');

coder.extrinsic('py.numpy.array');

coder.extrinsic('py.PyModelMatlab.predict'); % 如果predict是直接调用的Python函数，也需要声明为extrinsic



        在simulink中使：MATLAB function定义输入数据（test\_data）输入预测模块，再用一个MATLAB function调用python下的深度学习模型，这里特别需要注意的是，提前定义好预测输出的维度和数据类型，simulink的测试模型如下：

 其中中间的MATLAB Function就是调用深度学习模型的模块完整代码如下：

function y = fcn(u)

% 声明外部函数，告诉代码生成器不需要为其生成C代码

coder.extrinsic('py.importlib.import\_module');

coder.extrinsic('py.importlib.reload');

coder.extrinsic('py.numpy.array');

coder.extrinsic('py.PyModelMatlab.predict'); % 如果predict是直接调用的Python函数，也需要声明为extrinsic

y = zeros(1, 3, 'double');%% 预分配输出变量并明确指定类型和大小

% 初始化Python模块（只在首次调用时执行）

persistent predict\_module;

if isempty(predict\_module)

% 设置环境变量

setenv('KMP\_DUPLICATE\_LIB\_OK', 'TRUE');

% 导入Python模块

predict\_module = py.importlib.import\_module('PyModelMatlab');

py.importlib.reload(predict\_module);

end

% 确保输入u是正确的维度

if size(u, 1) > 1 && size(u, 2) == 15

u = u'; % 如果u是列向量，转置为行向量

end

% 显式转换为NumPy数组并指定数据类型

u\_np = py.numpy.array(u, 'float64');

y\_np = py.PyModelMatlab.predict(u\_np);

y = double(y\_np);



上述图片中就是正常运行的预测结果了

​