第一次小作业：乘法器

521030910376 应之未

在本次小作业中，我主要参考了 jittor 官方提供的计图文档库。我根据多层感知机在 PyTorch 中的代码实现，在该文档库中寻找对应的函数进行替代。计图文档库链接可点击以下文字访问：计图文档库链接。作业代码也可以在本链接中找到。

1. 数据集的生成

我采用 np.random.rand() 函数产生随机数。按照乘法器的需求，我生成了 5000 组 0 1 之间的随机数（1 组 2 个），并计算它们的乘积作为这一组的标签，其中前 4000 组作为训练集，后 1000 组作为测试集。

2. 多层感知机

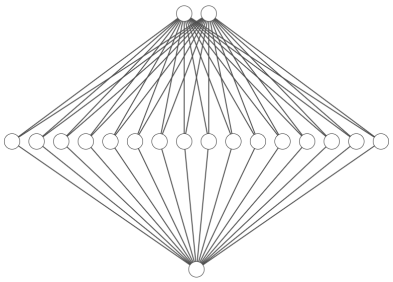

图 1: 多层感知机结构图

我采用的多层感知机结构为一层 2 节点的输入层，一层 16 节点的隐藏层和一层输出层。输入层和隐藏层全连接，隐藏层经过激活函数 relu() 后与输出层全连接。

3. 训练与参数选择

在本次作业中，我选用了 SGD 优化器，并选取均方误差作为损失函数，相关参数如下：

表 1: 参数

参数名	参数值
learning_rate	0.01
num_epochs	100000
momentum	0.9

4. 在测试集上的效果

为了更直观地展示模型在测试集上的效果，在测试过程中，我既输出了在测试集上的均方误差 (MSELoss) 和 L1 范数误差 (L1Loss)，也将测试集的标签和预测值在图像上进行表示。

得到的 L1Loss 为 0.013243，MSELoss 为 0.000286，运行结果如图 2。

```
jt.Var([0.013243], dtype=float32)
Test Loss: 0.00028606996056623757
```

图 2: 运行结果-测试集上的误差

将预测结果以图像方式呈现可以得到图 3：

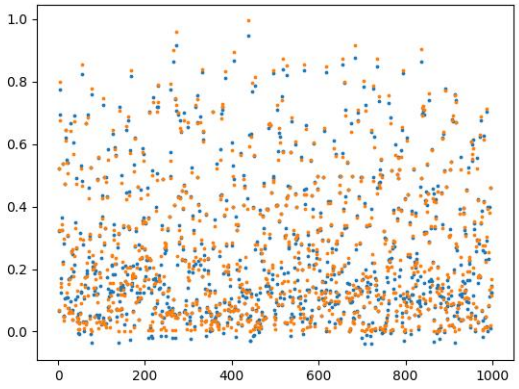

图 3: 运行结果

5. 对本作业的思考

神经网络中线性操作和非线性激活函数的组合可以使得神经网络具有表示复杂函数的能力。从实验结果中我们可以看出，尽管单个线性层无法直接拟合乘法，但当我们提高神经网络的层数，更多的线性函数和非线性函数的组合可以去逼近乘法操作，而且在训练数据范围内具有较好的准确性。