```
import torch

torch.manual_seed(0)

def relu(x):
    return torch.maximum(torch.tensor(0.0), x)

def sigmoid(x):
    return 1 / (1 + torch.exp(-x))

def tanh(x):
    return torch.tanh(x)

class Layer:
    def __init__(self, input_size, output_size, activation=None):
        self.w = torch.randn(input_size, output_size, requires_grad=True)
        self.b = torch.randn(output_size, requires_grad=True)
        self.activation = activation

    def forward(self, x):
        z = x @ self.w + self.b
        if self.activation == "relu":
            return relu(z)
        elif self.activation == "sigmoid":
            return sigmoid(z)
        elif self.activation == "tanh":
            return tanh(z)
        else:
            return z

class SimpleGraph:
    def __init__(self):
        self.layer1 = Layer(2, 3, activation="relu")
        self.layer2 = Layer(3, 2, activation="sigmoid")
        self.output_layer = Layer(1, 1, activation=None)

    def forward(self, x):
        a1 = self.layer1.forward(x)
        a2 = self.layer2.forward(a1)
        combined = a1.sum(dim=1) + a2.sum(dim=1)
        a3 = tanh(combined)
        a3 = a3.unsqueeze(1)
        output = self.output_layer.forward(a3)
        return output

x = torch.tensor([
    [1.0, 2.0],
    [2.0, -1.0],
    [0.5, 0.7],
    [3.0, 1.5]
], requires_grad=True)

net = SimpleGraph()
output = net.forward(x)
output.sum().backward()
```