**Sequential**

class torch.nn.Sequential

A sequential container.

Modules will be added to it in the order they are passed in the constructor. Alternatively, an OrderedDict of modules can be passed in. The forward() method of Sequential accepts any input and forwards it to the first module it contains. It then “chains” outputs to inputs sequentially for each subsequent module, finally returning the output of the last module.

The value a Sequential provides over manually calling a sequence of modules is that it allows treating the whole container as a single module, such that performing a transformation on the Sequential applies to each of the modules it stores (which are each a registered submodule of the Sequential).

What’s the difference between a Sequential and a torch.nn.ModuleList? A ModuleList is exactly what it sounds like–a list for storing Modules! On the other hand, the layers in a Sequential are connected in a cascading way.

Example:

# Using Sequential to create a small model. When `model` is run,

# input will first be passed to `Conv2d(1,20,5)`. The output of

# `Conv2d(1,20,5)` will be used as the input to the first

# `ReLU`; the output of the first `ReLU` will become the input

# for `Conv2d(20,64,5)`. Finally, the output of

# `Conv2d(20,64,5)` will be used as input to the second `ReLU`

model = nn.Sequential(

nn.Conv2d(1,20,5),

nn.ReLU(),

nn.Conv2d(20,64,5),

nn.ReLU()

)

# Using Sequential with OrderedDict. This is functionally the

# same as the above code

model = nn.Sequential(OrderedDict([

('conv1', nn.Conv2d(1,20,5)),

('relu1', nn.ReLU()),

('conv2', nn.Conv2d(20,64,5)),

('relu2', nn.ReLU())

]))