Tensorflow 2.0

简明实战教程

讲师：日月光华

Graph Execution 模式

讲师：日月光华　tf2.0 答疑群：738790253

图运算模式

在 TensorFlow 2.0 中，我们可以使用

@tf.function 装饰器

实现 Graph Execution，从而将模型转换为易于部署且高性能的 TensorFlow 图模型。

图运算

@tf.function装饰器

使用静态编译将函数内的代码转换成计算图

图运算

@tf.function 对函数内可使用的语句有一定限制（仅支持 Python 语言的一个子集），且需要函数内的操作本身能够被构建为计算图。

图运算

建议在函数内只使用 TensorFlow 的原生操作，不要使用过于复杂的 Python 语句，函数参数只包括 TensorFlow 张量或 NumPy 数组，并最好是能够按照计算图的思想去构建函数

图运算

@tf.function 可以带来了一定的性能提升。

一般，当模型由较多小的操作组成的时候，

@tf.function 带来的提升效果较大。

图运算

@tf.function 可以带来了一定的性能提升。

当模型的操作数量较少，但单一操作均很耗时的时候，则 @tf.function 带来的性能提升不会太大。

@tf.function 内在机制

当被 @tf.function 修饰的函数第一次被调用的时候，

进行以下操作：

@tf.function 内在机制

（1）在 Eager Execution 模式关闭的环境下，函数内的代码依次运行。也就是说，每个 tf. 方法都只是定义了计算节点，而并没有进行任何实质的计算。这与 TensorFlow 1.X 的 Graph Execution 是一致的

@tf.function 内在机制

（2）使用 AutoGraph 将函数中的 Python 控制流语句转换成 TensorFlow 计算图中的对应节点

（比如说 while 和 for 语句转换为 tf.while ， if 语句转换为 tf.cond 等等；

@tf.function 内在机制

（3）基于上面的两步，建立函数内代码的计算图表示（为了保证图的计算顺序，图中还会自动加入一些 tf.control_dependencies 节点）

@tf.function 内在机制

（4）运行一次这个计算图

（5）基于函数的名字和输入的函数参数的类型生成一个哈希值，并将建立的计算图缓存到一个哈希表中。

@tf.function 内在机制

（6）在被 @tf.function 修饰的函数之后再次被调用的时候，根据函数名和输入的函数参数的类型计算哈希值，检查哈希表中是否已经有了对应计算图的缓存。如果是，则直接使用已缓存的计算图，否则重新按上述步骤建立计算图。

@tf.function 内在机制

总结：

AutoGraph 起到了类似编译器的作用，能够帮助我们通过更加自然的 Python 控制流轻松地构建带有条件 / 循环的计算图，而无需手动使用 TensorFlow 的 API 进行构建。

@tf.function 使用

当定义多个函数实现不同的运算式，

仅需要在最后调用的函数上添加@tf.function装饰器

即可

这样所有的运算节点都会被编译

日月光华

日月光华网易云课堂

日月光华微信

谢谢大家

讲师：日月光华　　tf2.0 答疑群：738790253