# BKM Formalism: TensorFlow Graph Execution and tf.function

Cole Weis - 12/21/21

## Graph Execution vs. Eager Execution

**Eager Execution**

- Expressions evaluate as if executed exactly in order, one at a time
  - In python they are always executed explicitly in order

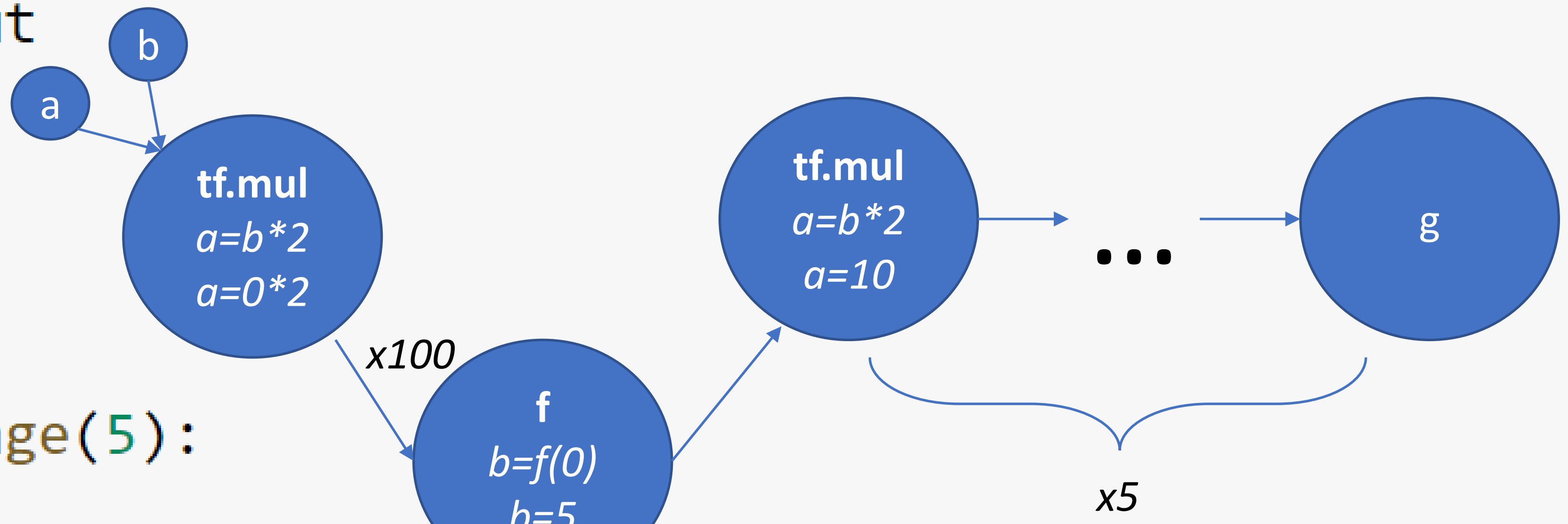
**Graph Execution**

- Functions are localized, with more defined inputs and outputs
- Allows for graph optimization of function dependencies and lazy execution
- Much more efficient for lots of small operations
- Better optimized for use with TPUs (Tensor Processing Units)

```python
def f(c):
  time.sleep(10e5) # -> Computation that takes some time
  output = c + 5
  return output


def g():
  a, b = 0
  for i in range(5):
    a = b * 2
    for j in range(100):
      b = f(a)
  return a


g()
```

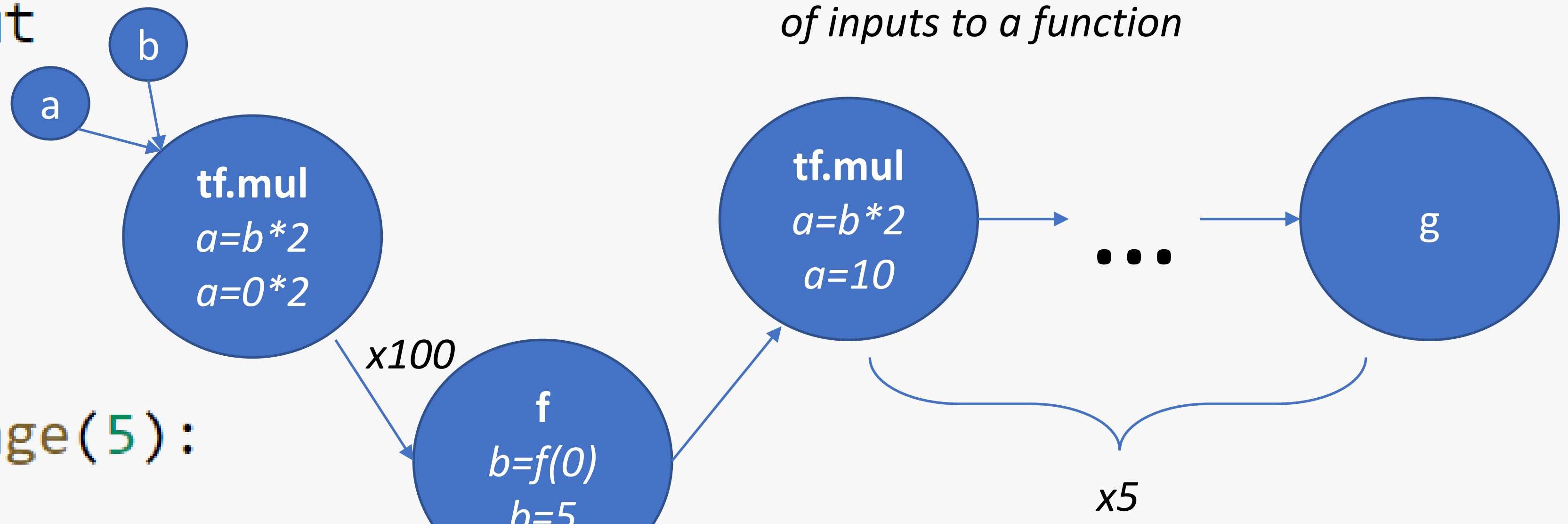
**f() Runs 500 times**

```
@tf.function
def f(c):
  time.sleep(10e5) # -> Computation that takes some time
  output = c + 5
  return output

@tf.function
def g():
  a, b = 0
  for i in range(5):
    a = b * 2
    for j in range(100):
      b = f(a)
  return a

g()
```



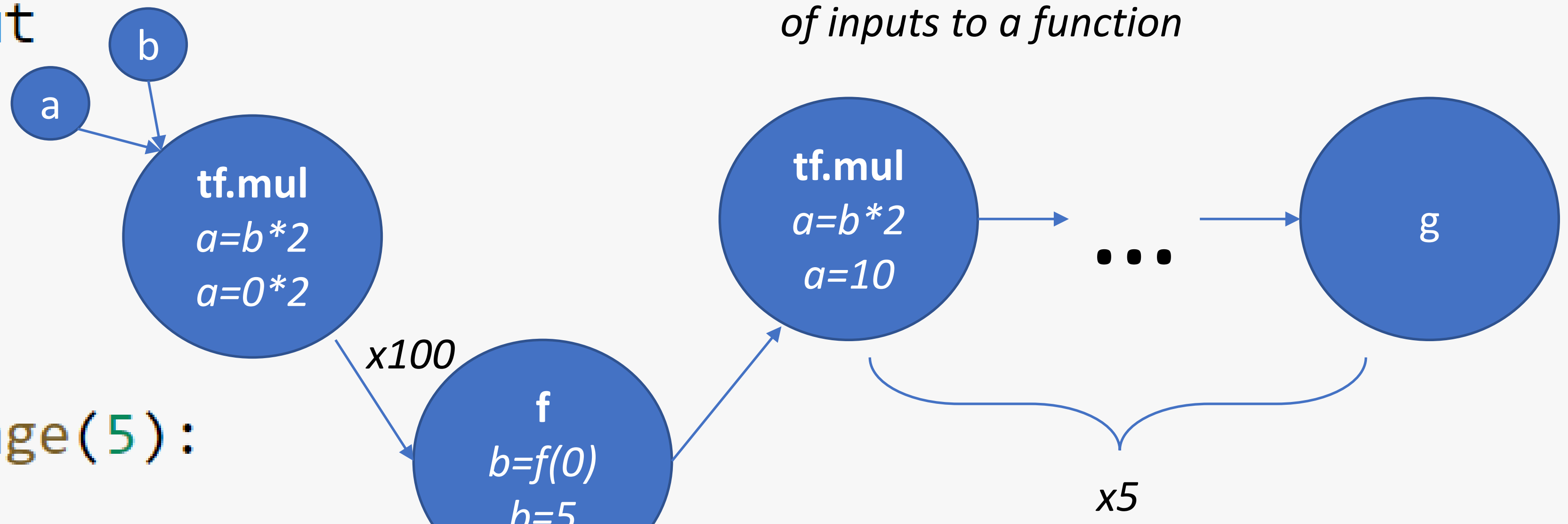
```
@tf.function
def f(c):
  time.sleep(10e5) # -> Computation that takes some time
  output = c + 5
  return output


@tf.function
def g():
  a, b = 0
  for i in range(5):
    a = b * 2
    for j in range(100):
      b = f(a)
  return a


g()
```

*-> Graph is only retraced for each different set of inputs to a function*



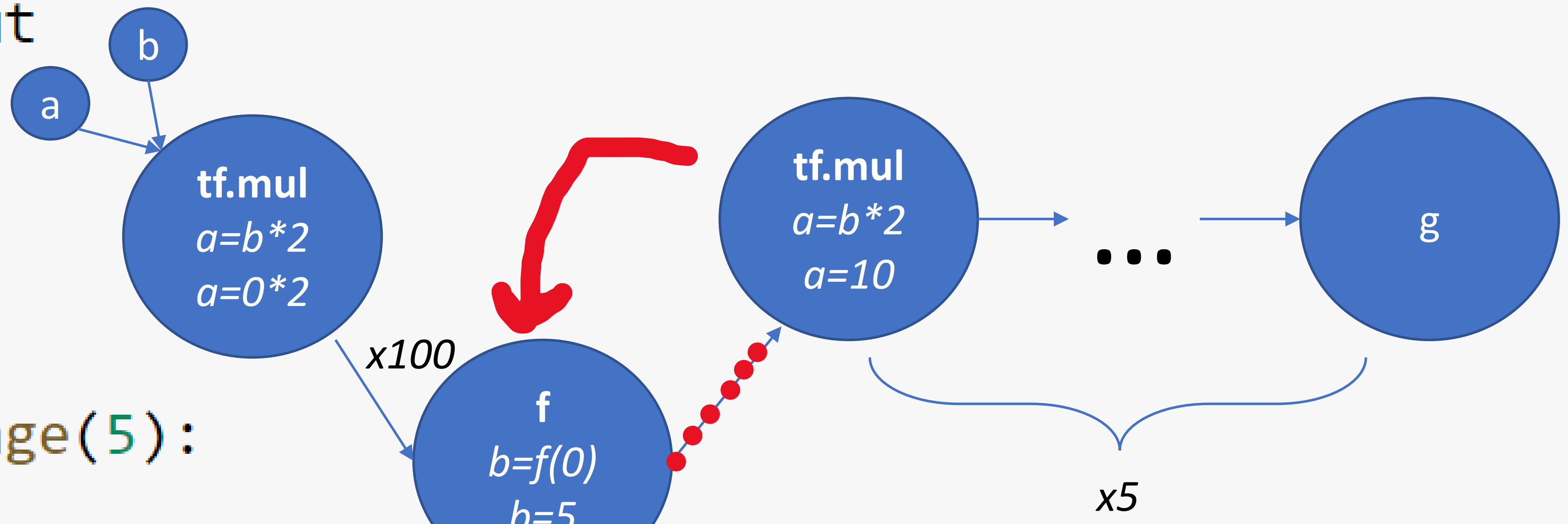
```python
@tf.function
def f(c):
  time.sleep(10e5) # -> Computation that takes some time
  output = c + 5
  return output

@tf.function
def g():
  a, b = 0
  for i in range(5):
    a = b * 2
    for j in range(100):
      b = f(a)
  return a

g()
```

*-> Graph is only retraced for each different set of inputs to a function*



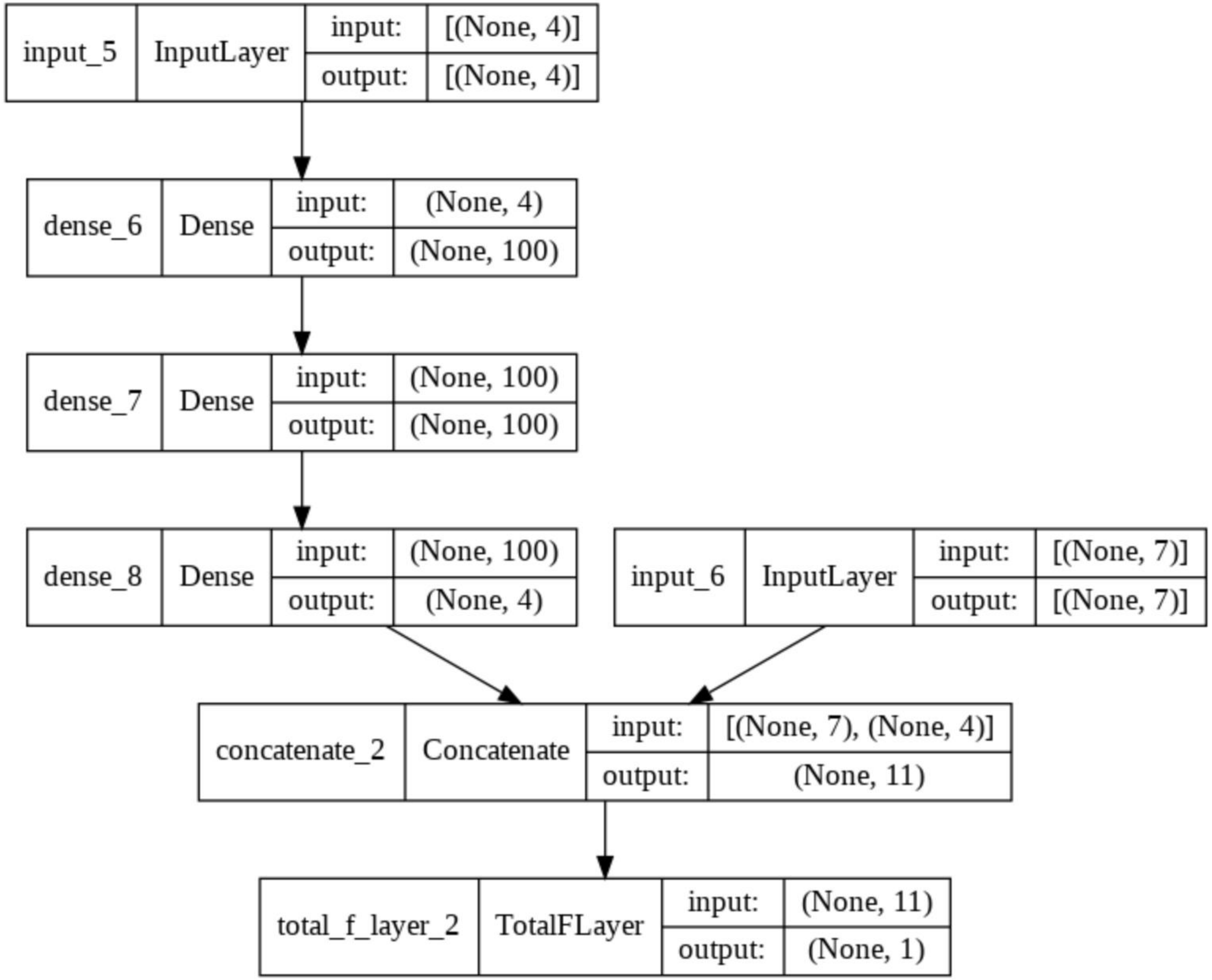
**f() Runs 5 times**

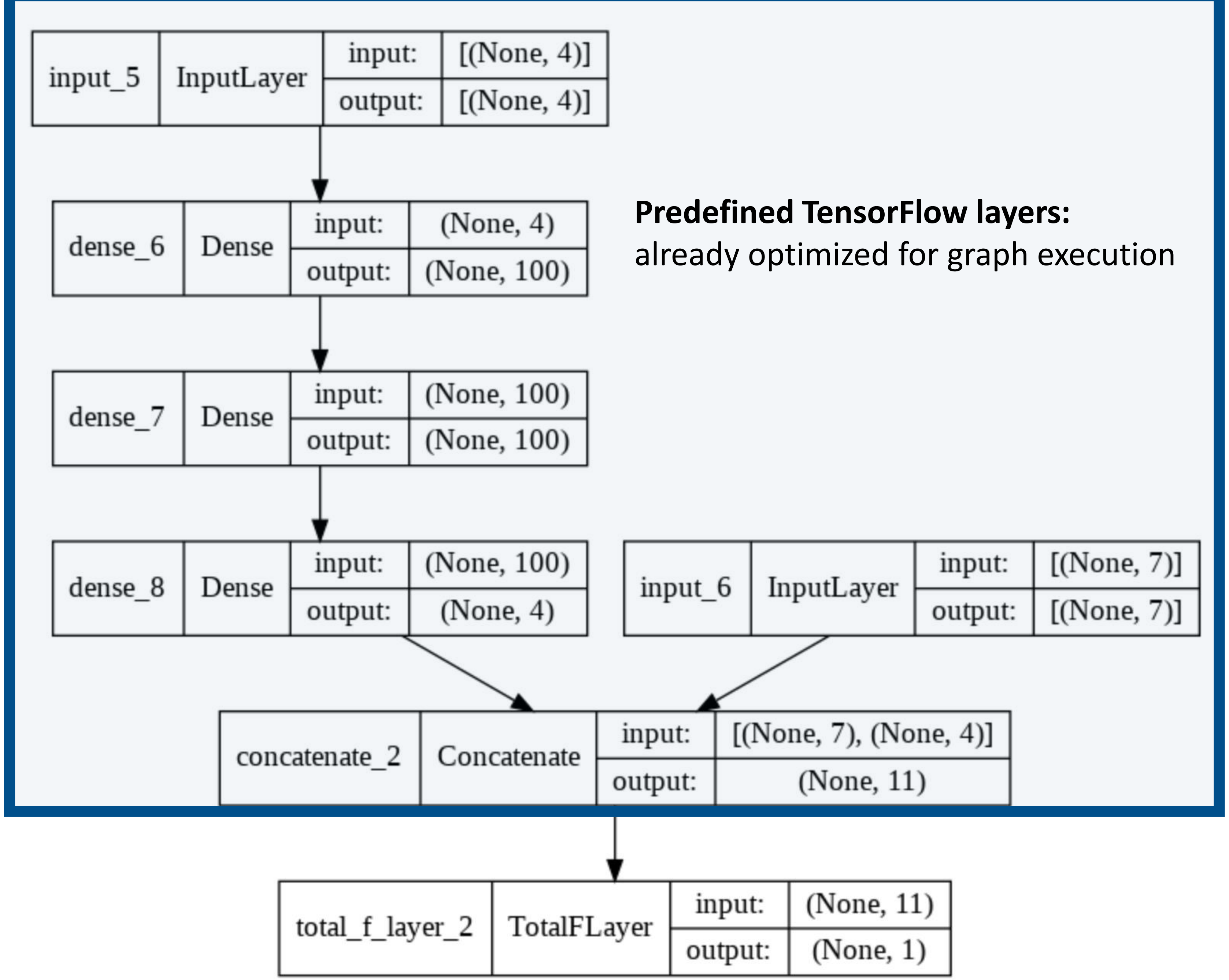
```
@tf.function
def f(c):
  time.sleep(10e5) # -> Computation that takes some time
  output = c + 5
  return output

@tf.function
def g():
  a, b = 0
  for i in range(5):
    a = b * 2
    for j in range(100):
      b = f(a)
  return a

g()
```



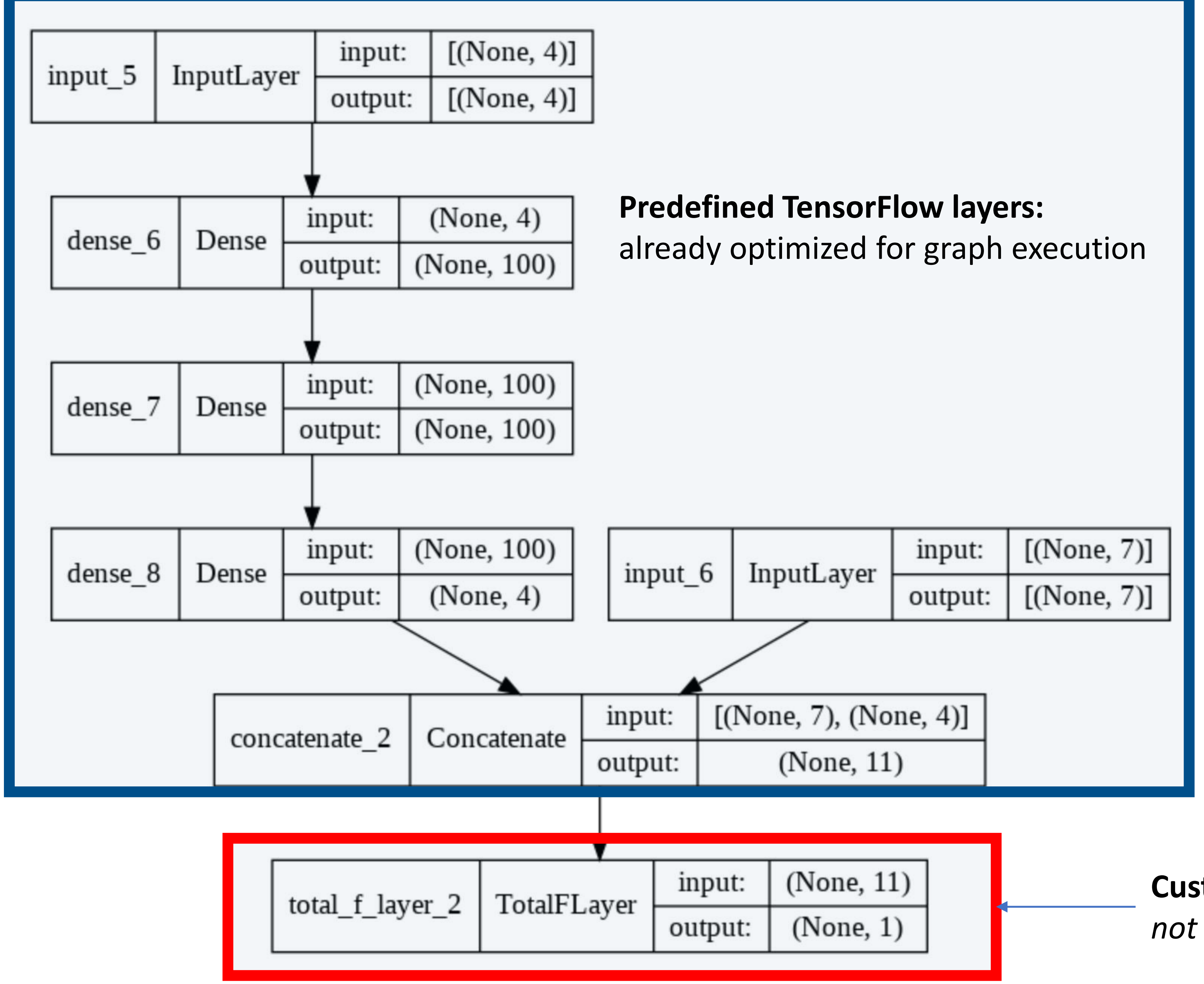
*-> Allows for Lazy Execution*

## Local Fit Model


input_5 | InputLayer | input: [(None, 4)] | output: [(None, 4)]
dense_6 | Dense | input: (None, 4) | output: (None, 100)
dense_7 | Dense | input: (None, 100) | output: (None, 100)
dense_8 | Dense | input: (None, 100) | output: (None, 4)
input_6 | InputLayer | input: [(None, 7)] | output: [(None, 7)]
concatenate_2 | Concatenate | input: [(None, 7), (None, 4)] | output: (None, 11)
total_f_layer_2 | TotalFLayer | input: (None, 11) | output: (None, 1)



**Predefined TensorFlow layers:**
already optimized for graph execution



**Predefined TensorFlow layers:**
already optimized for graph execution

**Custom defined layer:**
*not* optimized for graph execution

## Graph Execution vs. Eager Execution

**Restrictions for Graph Execution of a function (@tf.function):**

1. Replace Pythonic Expressions with tensorflow equivalents
2. All inputs and outputs are tensors
3. Can only modify local method variables and outputs

*Defining input shape of tensors is helpful for increasing efficacy*

## Refactoring TotalF Layer for Graph Execution

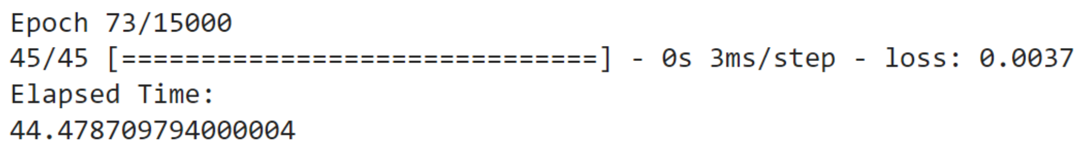
Localizing Methods, Calling all methods externally, Passing tensors instead of python variables

# Results

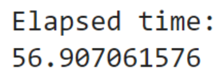
**Eager Execution**

**Graph Execution**

5 Sets

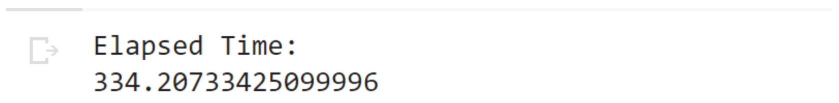


```
Epoch 73/15000
45/45 [==============================] - 0s 3ms/step - loss: 0.0037
Elapsed Time:
44.478709794000004
```

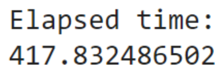


```
Elapsed time:
56.907061576
```

50 Sets



```
Elapsed Time:
334.20733425099996
```

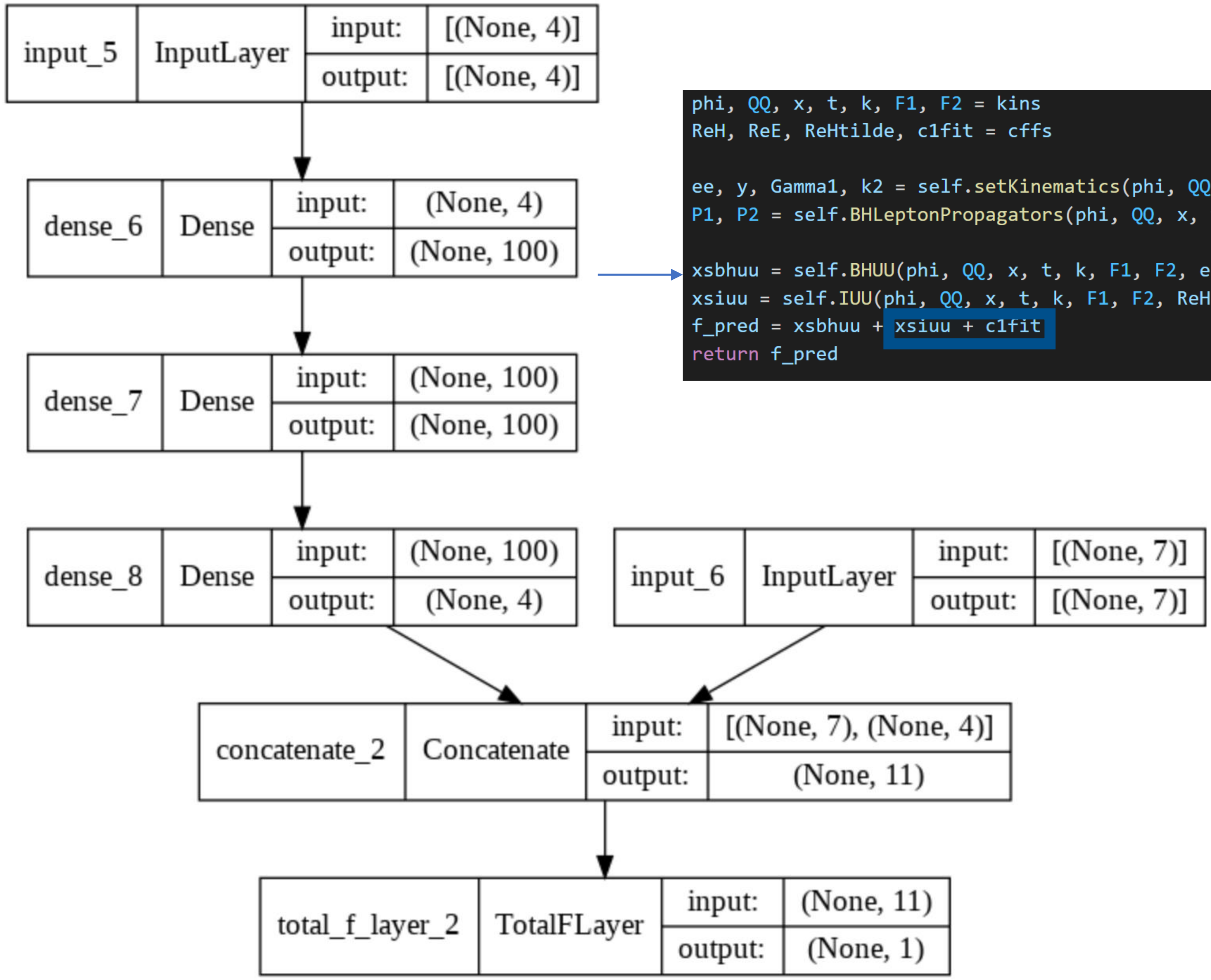


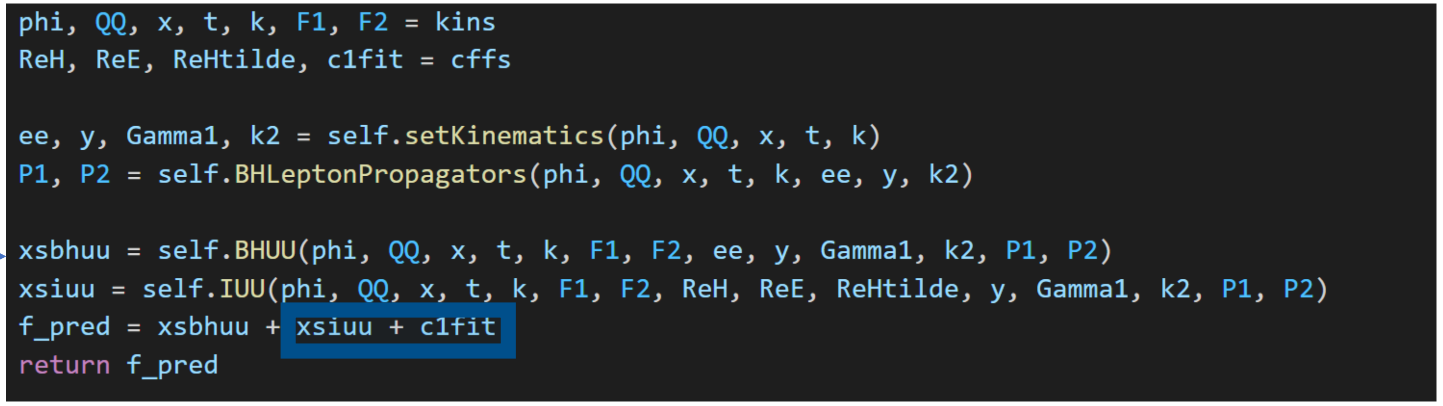
```
Elapsed time:
417.832486502
```

## **Problem:** Excessive Graph Retracing

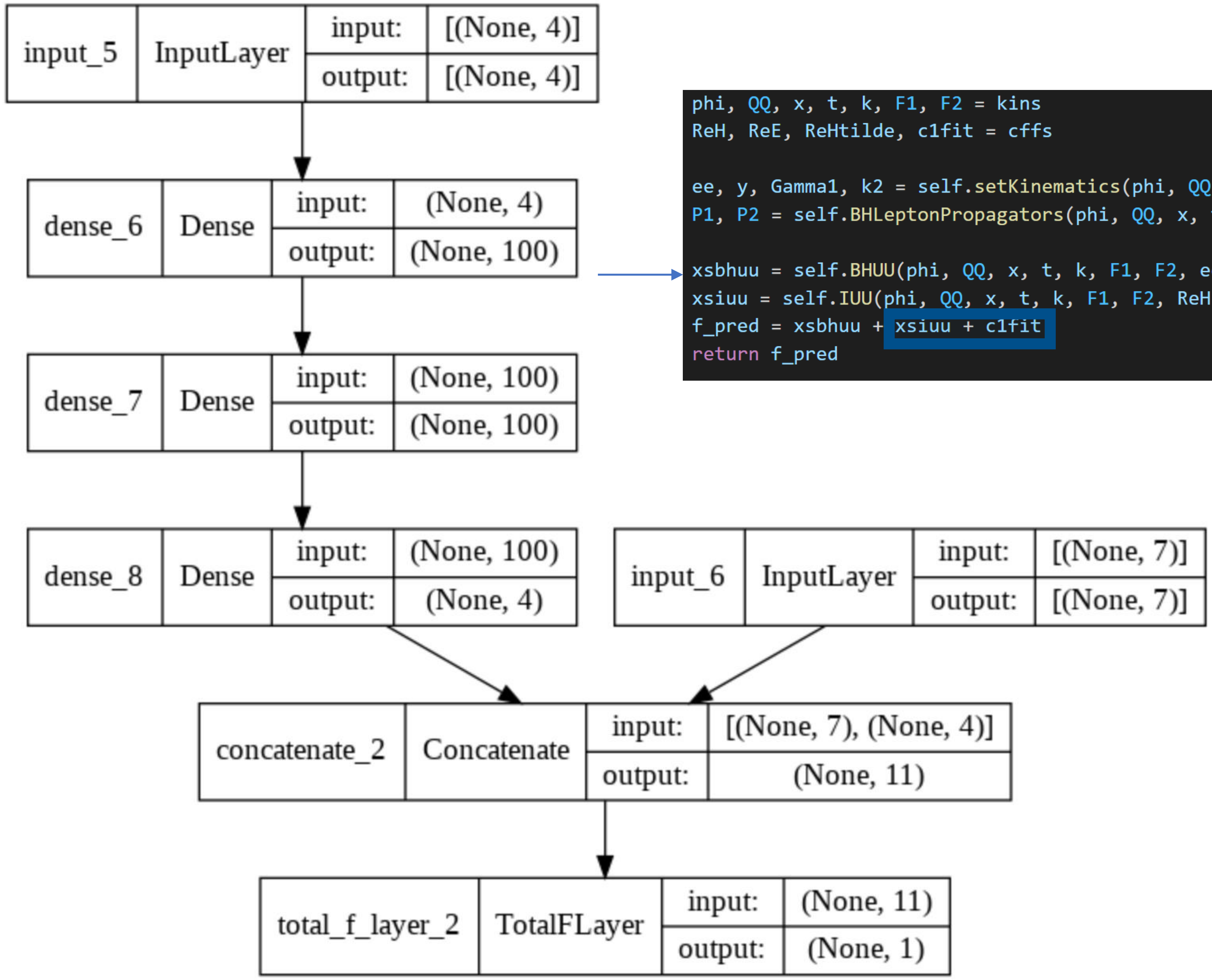
```
WARNING:tensorflow:6 out of the last 6 calls to <function BHDVCStf.setKinematics at 0x7fe04731c8c0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings cou
WARNING:tensorflow:6 out of the last 6 calls to <function BHDVCStf.BHLeptonPropagators at 0x7fe043198950> triggered tf.function retracing. Tracing is expensive and the excessive number of tracin
WARNING:tensorflow:6 out of the last 6 calls to <function BHDVCStf.BHUU at 0x7fe047407680> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due
WARNING:tensorflow:6 out of the last 6 calls to <function BHDVCStf.IUU at 0x7fe047407f80> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due
```

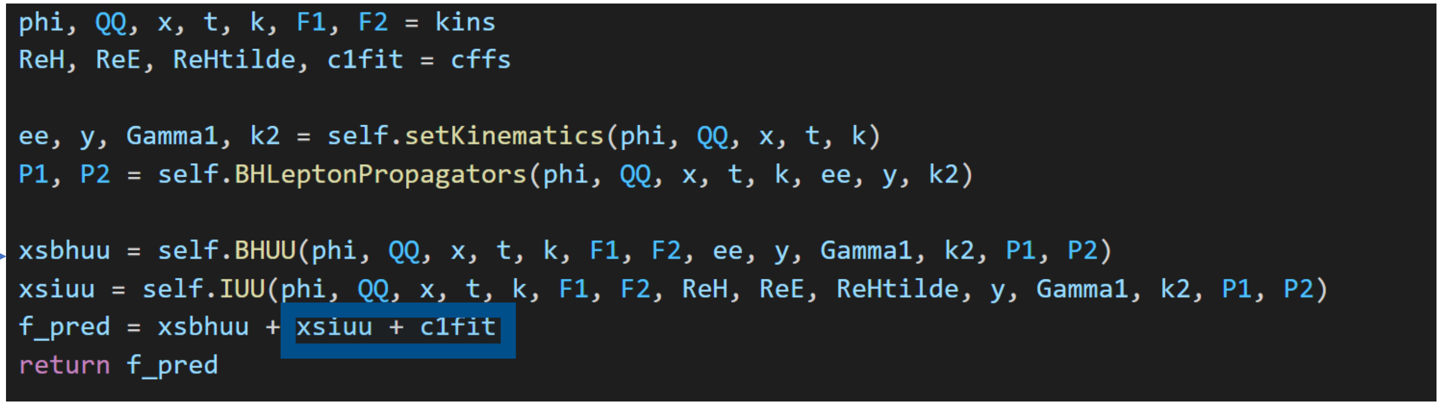
```
tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors.
```


input_5 | InputLayer | input: [(None, 4)] | output: [(None, 4)]
dense_6 | Dense | input: (None, 4) | output: (None, 100)
dense_7 | Dense | input: (None, 100) | output: (None, 100)
dense_8 | Dense | input: (None, 100) | output: (None, 4)
input_6 | InputLayer | input: [(None, 7)] | output: [(None, 7)]
concatenate_2 | Concatenate | input: [(None, 7), (None, 4)] | output: (None, 11)
total_f_layer_2 | TotalFLayer | input: (None, 11) | output: (None, 1)



```
phi, QQ, x, t, k, F1, F2 = kins
ReH, ReE, ReHtilde, c1fit = cffs

ee, y, Gamma1, k2 = self.setKinematics(phi, QQ, x, t, k)
P1, P2 = self.BHLeptonPropagators(phi, QQ, x, t, k, ee, y, k2)

xsbhuu = self.BHUU(phi, QQ, x, t, k, F1, F2, ee, y, Gamma1, k2, P1, P2)
xsiuu = self.IUU(phi, QQ, x, t, k, F1, F2, ReH, ReE, ReHtilde, y, Gamma1, k2, P1, P2)
f_pred = xsbhuu + xsiuu + c1fit
return f_pred
```


input_5 | InputLayer | input: [(None, 4)] | output: [(None, 4)]
dense_6 | Dense | input: (None, 4) | output: (None, 100)
dense_7 | Dense | input: (None, 100) | output: (None, 100)
dense_8 | Dense | input: (None, 100) | output: (None, 4)
input_6 | InputLayer | input: [(None, 7)] | output: [(None, 7)]
concatenate_2 | Concatenate | input: [(None, 7), (None, 4)] | output: (None, 11)
total_f_layer_2 | TotalFLayer | input: (None, 11) | output: (None, 1)



```
phi, QQ, x, t, k, F1, F2 = kins
ReH, ReE, ReHtilde, c1fit = cffs

ee, y, Gamma1, k2 = self.setKinematics(phi, QQ, x, t, k)
P1, P2 = self.BHLeptonPropagators(phi, QQ, x, t, k, ee, y, k2)

xsbhuu = self.BHUU(phi, QQ, x, t, k, F1, F2, ee, y, Gamma1, k2, P1, P2)
xsiuu = self.IUU(phi, QQ, x, t, k, F1, F2, ReH, ReE, ReHtilde, y, Gamma1, k2, P1, P2)
f_pred = xsbhuu + xsiuu + c1fit
return f_pred
```