Block Layer

Stack multiple blocks together to create a block layer in ResNet.

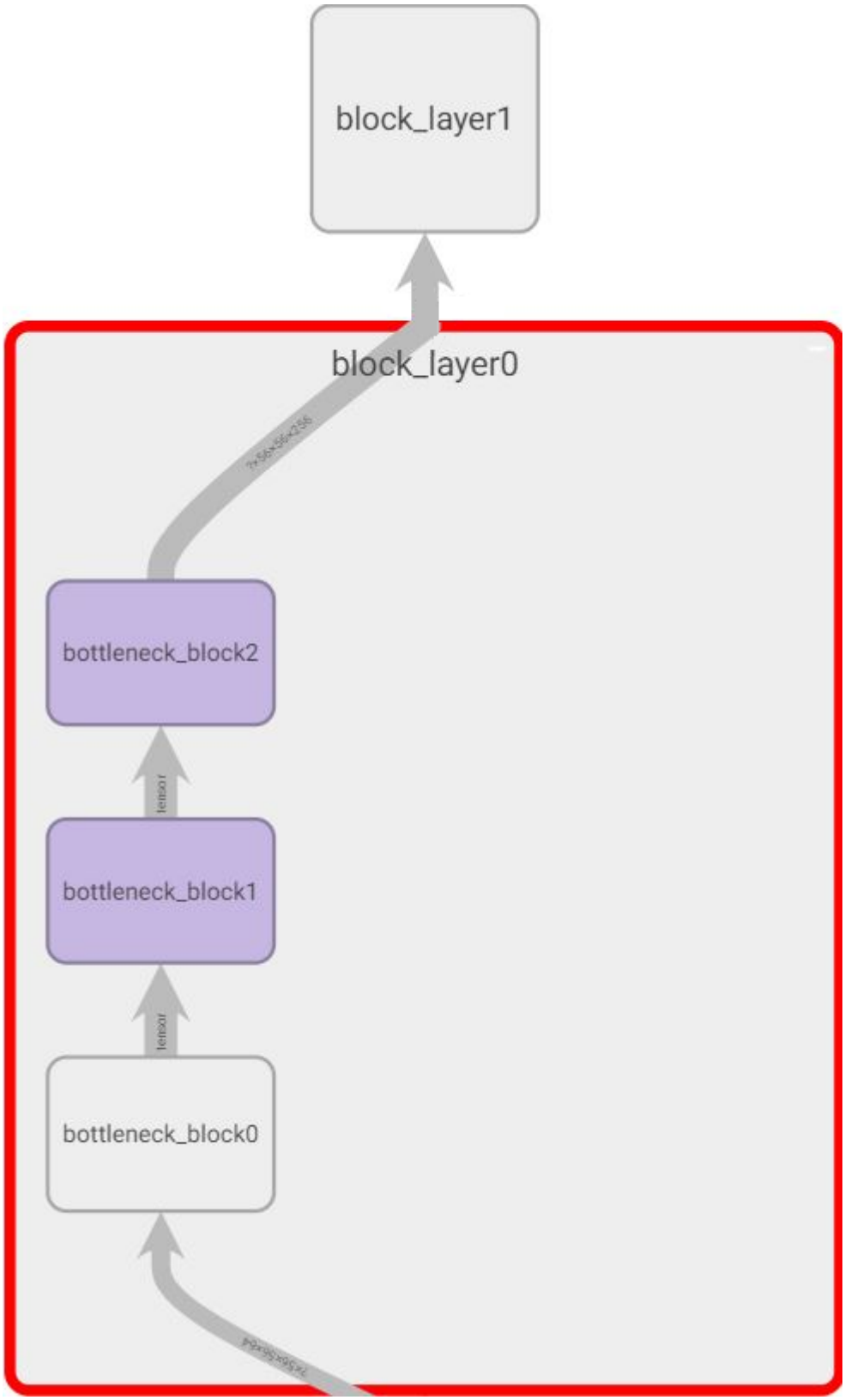
Chapter Goals:

- Learn how a layer of blocks is organized
- Understand the function for creating a layer of blocks

A. Block layers

A ResNet model is made up of four block layers. Each block layer will contain a different number of blocks, depending on the total number of weight layers in the ResNet model. For example, an 18 layer ResNet model has 2 blocks in each block layer.

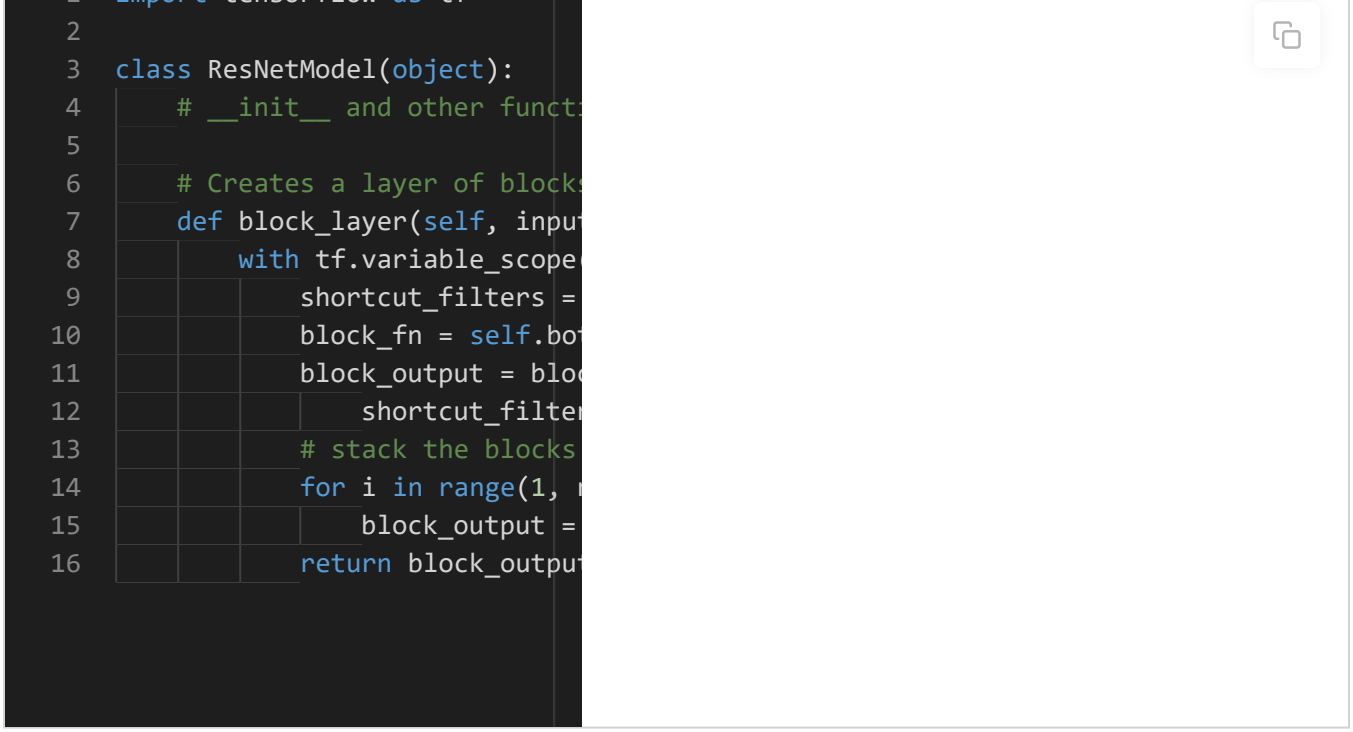
The blocks within each block layer are connected, so the output of block i is the input of block $i + 1$ in the same layer. Furthermore, the four block layers themselves are connected, so the output of block layer j becomes the input of block layer $j + 1$.



First block layer of a ResNet model with 50 weight layers. The first bottleneck block uses a projection shortcut, so it is colored differently. The output of the first block layer becomes the input of the second block layer.

The block layers account for a large majority of the weight layers in a ResNet model. However, they don't make up the entire structure of a ResNet model. The block layer is implemented in the `ResNetModel` class as the `block_layer` function:

```
1  import tensorflow as tf
```



```
import tensorflow as tf

class ResNetModel(object):
    # __init__ and other functi

    # Creates a layer of blocks
    def block_layer(self, input
        with tf.variable_scope(
            shortcut_filters =
            block_fn = self.bot
            block_output = bloc
                shortcut_filter
            # stack the blocks
            for i in range(1, r
                block_output =
            return block_output
```

As you can see, the function stacks multiple building blocks. The particular block to use (`block_fn`) depends on whether or not the model uses bottlenecks.