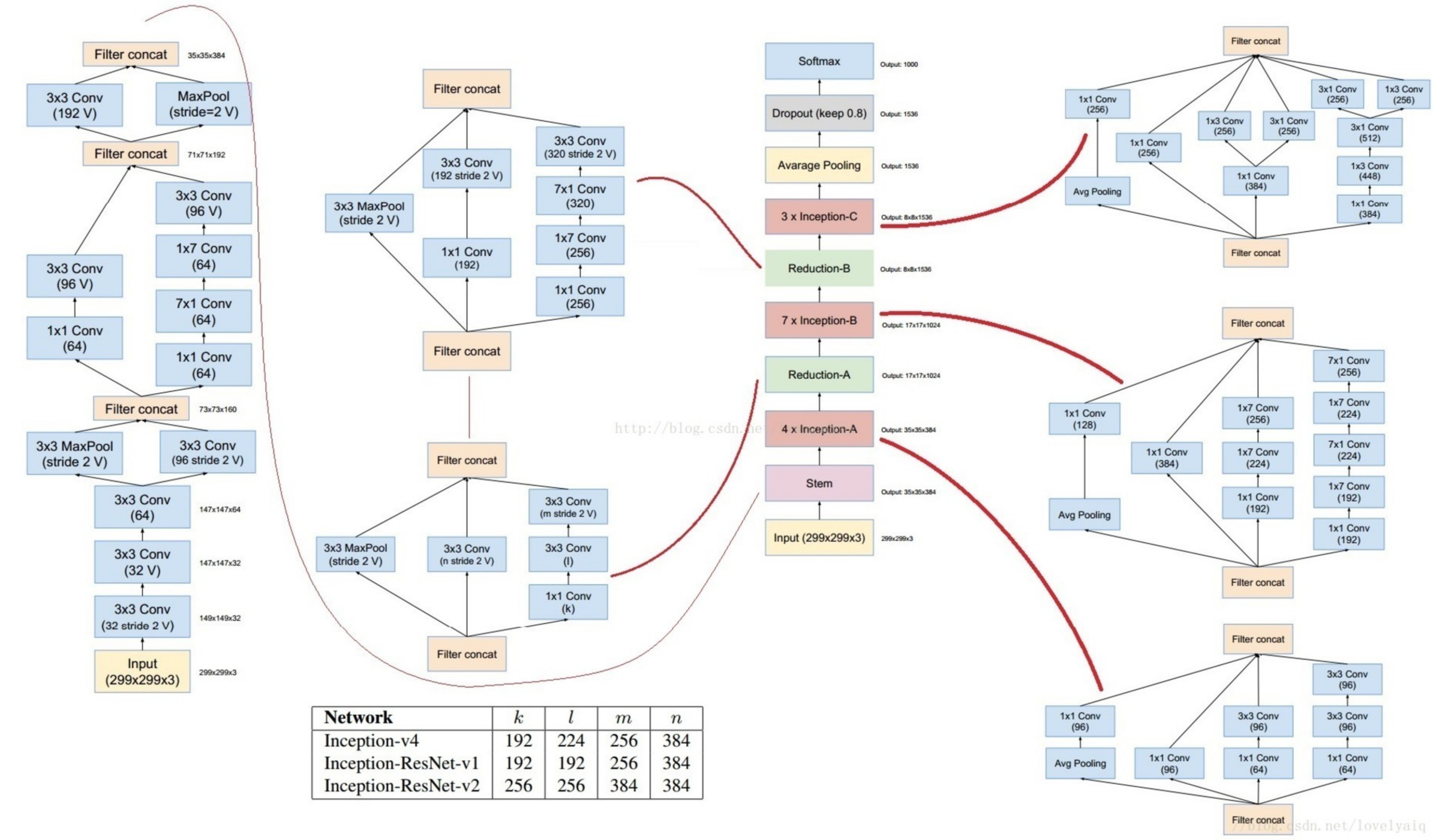
深度学习--Inception-ResNet-v1网络结构

原创 TiRan_Yang 于 2018-01-10 17:15:08 发布 阅读量2.9w 收藏 92 点赞数 16 版权
分类专栏： TensorFlow 文章标签： Inception-ResNet-v1 Tensorflow facenet

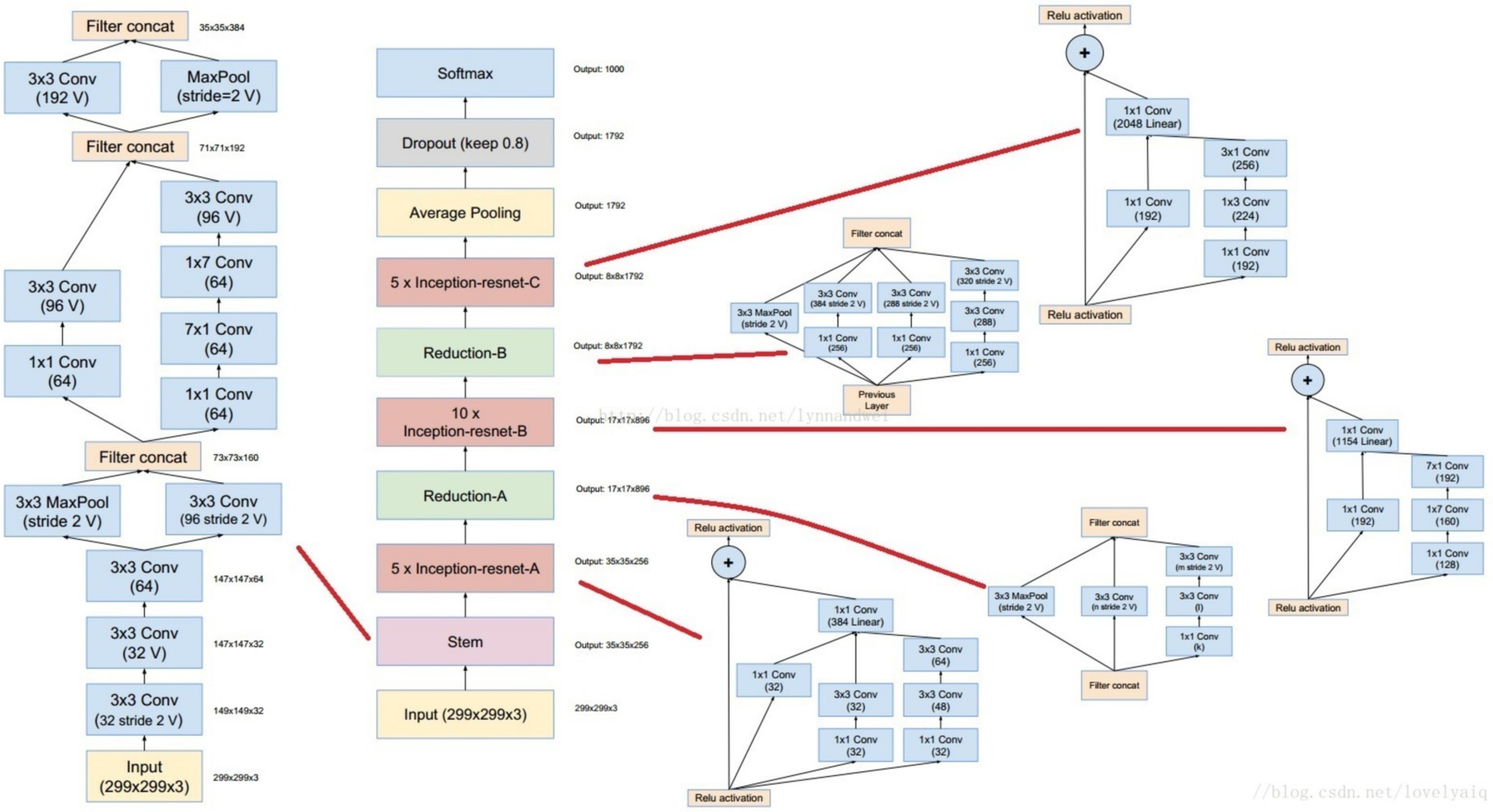
Inception V4的网络结构如下：



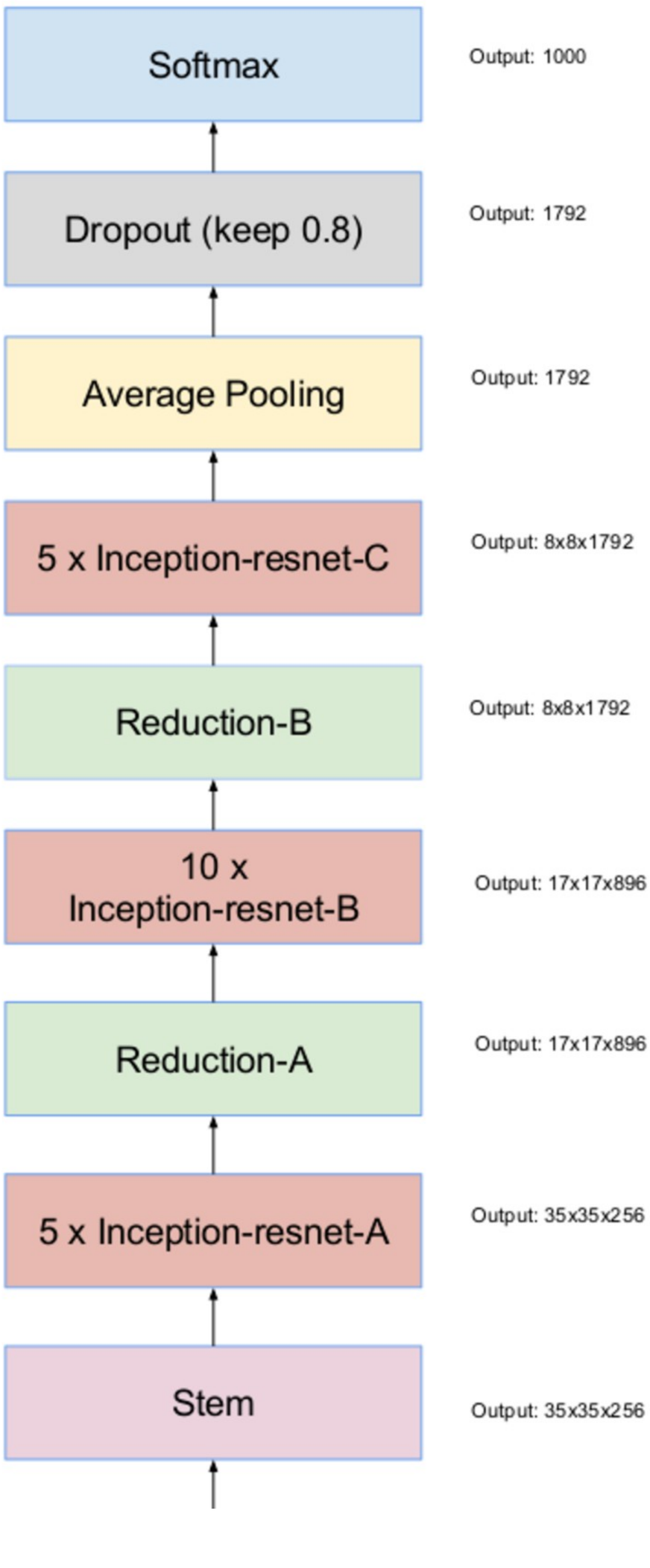
Network	k	l	m	n
Inception-v4	192	224	256	384
Inception-ResNet-v1	192	192	256	384
Inception-ResNet-v2	256	256	384	384

从图中可以看出，输入部分与V1到V3的输入部分有较大的差别，这样设计的目的为了：使用并行结构、不对称卷积核结构，可以在保证信息损失足够小的情况下，降低计算量。结构中1*1的卷积核也用来降维，并且也增加了非线性。

Inception-ResNet-v2与Inception-ResNet-v1的结构类似，除了stem部分。Inception-ResNet-v2的stem与V4的结构类似，Inception-ResNet-v2的输出chnnel要高。Reduction-A相同，Inception-ResNet-A、Inception-ResNet-B、Inception-ResNet-C和Reduction-B的结构与v1的类似，只不过输出的channel数量更多。



Inception-ResNet-v1的总体网络结构如下所示：



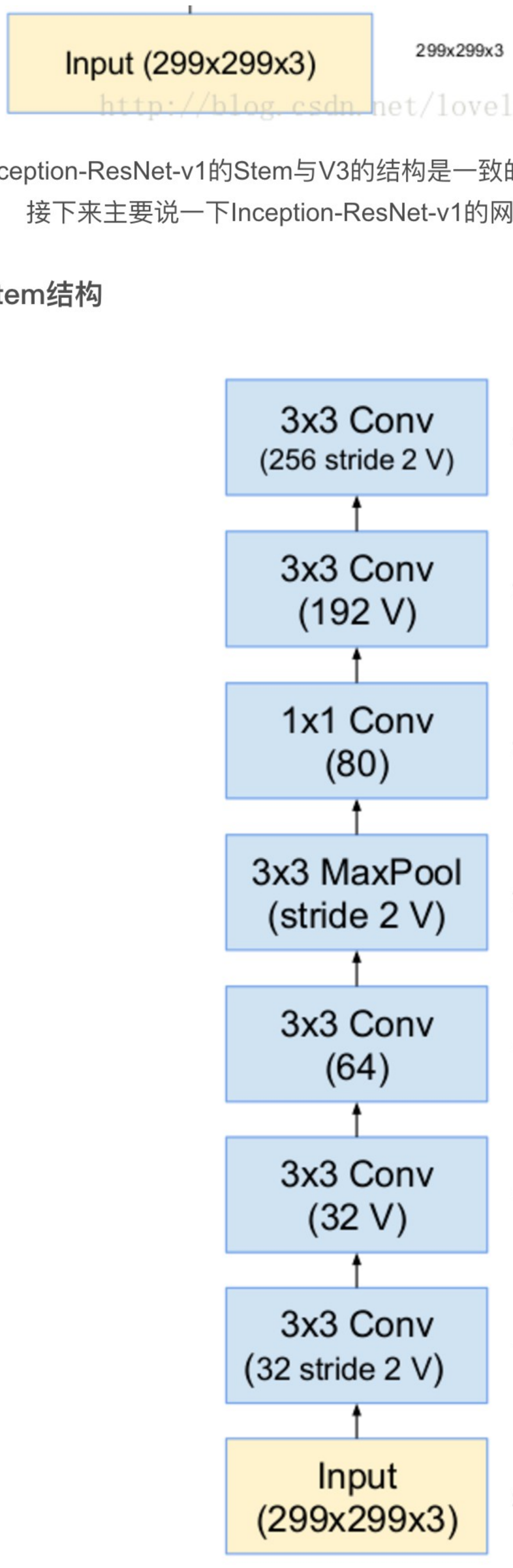


Figure 14. The stem of the Inception-ResNet-v1 network.

Inception-ResNet-v1的Stem与V3的结构是一致的。

接下来主要说一下Inception-ResNet-v1的网络结构及代码的实现部分。

Stem结构

stem结构与V3的Stem结构类似。

对应的代码为

```
with slim.arg_scope([slim.conv2d, slim.max_pool2d, slim.avg_pool2d],stride=1, padding='SAME'):

    # 149 x 149 x 32
    net = slim.conv2d(inputs, 32, 3, stride=2, padding='VALID', scope='Conv2d_1a_3x3')
    end_points['Conv2d_1a_3x3'] = net
    # 147 x 147 x 32
    net = slim.conv2d(net, 32, 3, padding='VALID',
scope='Conv2d_2a_3x3')
    end_points['Conv2d_2a_3x3'] = net
    # 147 x 147 x 64
    net = slim.conv2d(net, 64, 3, scope='Conv2d_2b_3x3')
end_points['Conv2d_2b_3x3'] = net
    # 73 x 73 x 64
    net = slim.max_pool2d(net, 3, stride=2, padding='VALID', scope='MaxPool_3a_3x3')
    end_points['MaxPool_3a_3x3'] = net
    # 73 x 73 x 80
    net = slim.conv2d(net, 80, 1, padding='VALID',
scope='Conv2d_3b_1x1')
    end_points['Conv2d_3b_1x1'] = net
    # 71 x 71 x 192
    net = slim.conv2d(net, 192, 3, padding='VALID',
scope='Conv2d_4a_3x3')
    end_points['Conv2d_4a_3x3'] = net
    # 35 x 35 x 256
    net = slim.conv2d(net, 256, 3, stride=2, padding='VALID',
scope='Conv2d_4b_3x3')
   end_points['Conv2d_4b_3x3'] = net
```

Inception-resnet-A模块

Inception-resnet-A模块是要重复5次的，网络结构为：

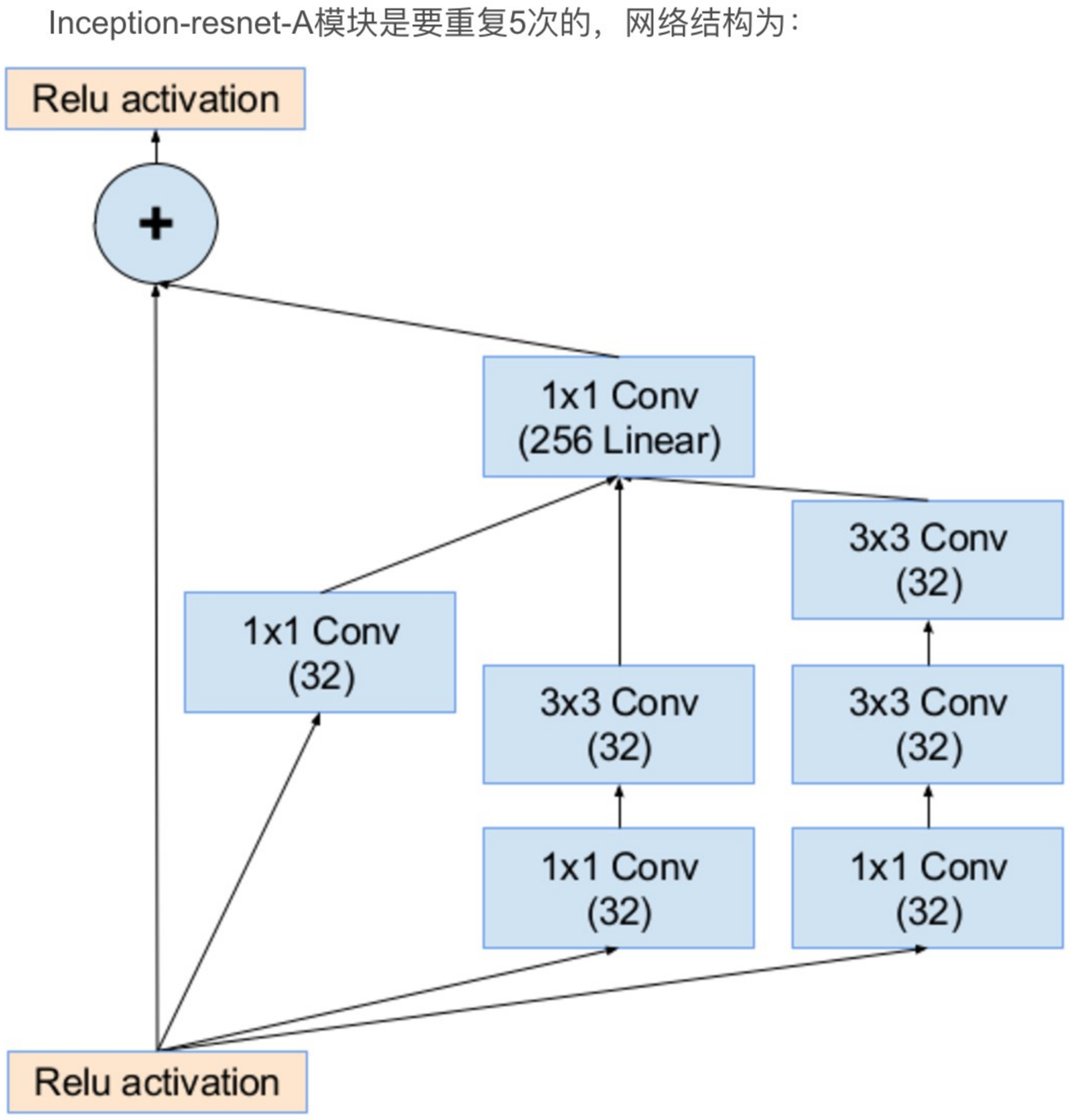


Figure 10. The schema for 35 × 35 grid (Inception-ResNet-A) module of Inception-ResNet-v1 network.

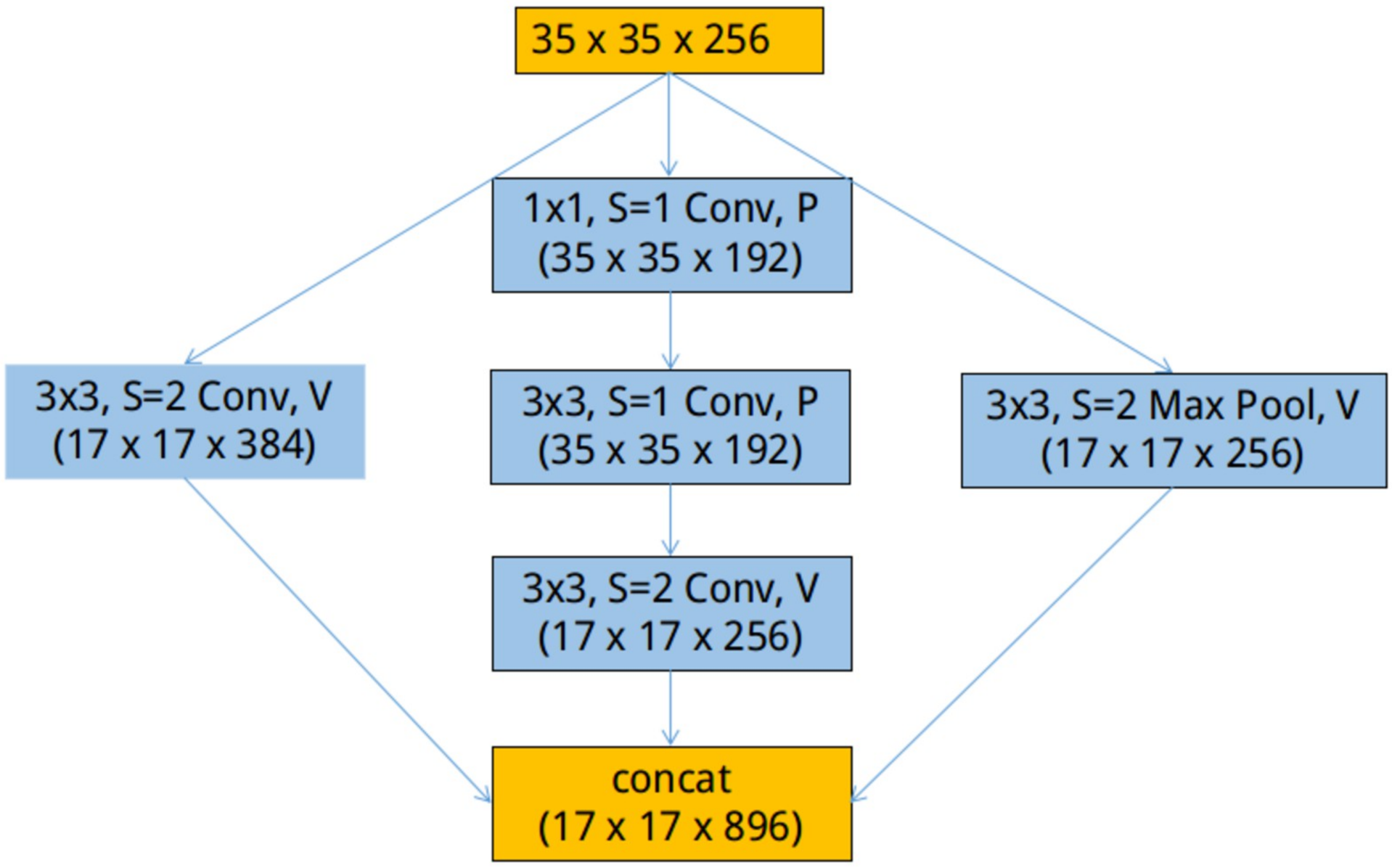
对应的代码表示为：

```python
# Inception-Renset-A
def block35(net, scale=1.0, activation_fn=tf.nn.relu, scope=None, reuse=None):
    """Builds the 35x35 resnet block."""
    with tf.variable_scope(scope, 'Block35', [net], reuse=reuse):
        with tf.variable_scope('Branch_0'):
            # 35 × 35 × 32
            tower_conv = slim.conv2d(net, 32, 1, scope='Conv2d_1x1')
        with tf.variable_scope('Branch_1'):
            # 35 × 35 × 32
            tower_conv1_0 = slim.conv2d(net, 32, 1, scope='Conv2d_0a_1x1')
            # 35 × 35 × 32
            tower_conv1_1 = slim.conv2d(tower_conv1_0, 32, 3, scope='Conv2d_0b_3x3')
        with tf.variable_scope('Branch_2'):
            # 35 × 35 × 32
            tower_conv2_0 = slim.conv2d(net, 32, 1, scope='Conv2d_0a_1x1')
            # 35 × 35 × 32
            tower_conv2_1 = slim.conv2d(tower_conv2_0, 32, 3, scope='Conv2d_0b_3x3')
            # 35 × 35 × 32
            tower_conv2_2 = slim.conv2d(tower_conv2_1, 32, 3, scope='Conv2d_0c_3x3')
        # 35 × 35 × 96
        mixed = tf.concat([tower_conv, tower_conv1_1, tower_conv2_2], 3)
        # 35 × 35 × 256
        up = slim.conv2d(mixed, net.get_shape()[3], 1, normalizer_fn=None,activation_fn=None, scope='Conv2
        # 使用残差网络scale = 0.17
        net += scale * up
        if activation_fn:
            net = activation_fn(net)
    return net

# 5 x Inception-resnet-A
net = slim.repeat(net, 5, block35, scale=0.17)
end_points['Mixed_5a'] = net
```

Reduction-A结构

Reduction-A中含有4个参数k、l、 m、 n，它们对应的值分别为：192, 192, 256, 384，在该层网络结构，输入为35×35×256，输出为17×17×896.

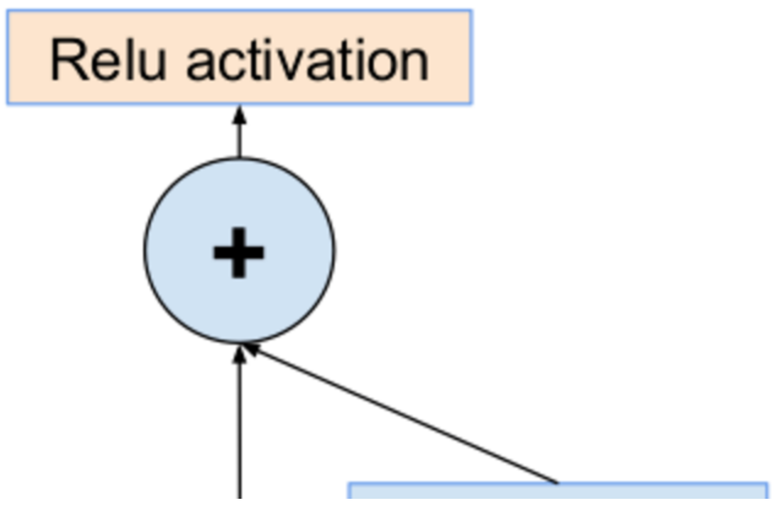


```python
def reduction_a(net, k, l, m, n):
    # 192, 192, 256, 384
    with tf.variable_scope('Branch_0'):
        # 17×17×384
        tower_conv = slim.conv2d(net, n, 3, stride=2, padding='VALID',
                                 scope='Conv2d_1a_3x3')
    with tf.variable_scope('Branch_1'):
        # 35×35×192
        tower_conv1_0 = slim.conv2d(net, k, 1, scope='Conv2d_0a_1x1')
        # 35×35×192
        tower_conv1_1 = slim.conv2d(tower_conv1_0, l, 3,
                                    scope='Conv2d_0b_3x3')
        # 17×17×256
        tower_conv1_2 = slim.conv2d(tower_conv1_1, m, 3,
                                    stride=2, padding='VALID',
                                    scope='Conv2d_1a_3x3')
    with tf.variable_scope('Branch_2'):
        # 17×17×256
        tower_pool = slim.max_pool2d(net, 3, stride=2, padding='VALID',
                                     scope='MaxPool_1a_3x3')
    # 17×17×896
    net = tf.concat([tower_conv, tower_conv1_2, tower_pool], 3)
    return net

# Reduction-A
with tf.variable_scope('Mixed_6a'):
    net = reduction_a(net, 192, 192, 256, 384)
    end_points['Mixed_6a'] = net
```

Inception-Resnet-B

Inception-Resnet-B模块是要重复10次，输入为17×17×896，输出为17×17×896，网络结构为：



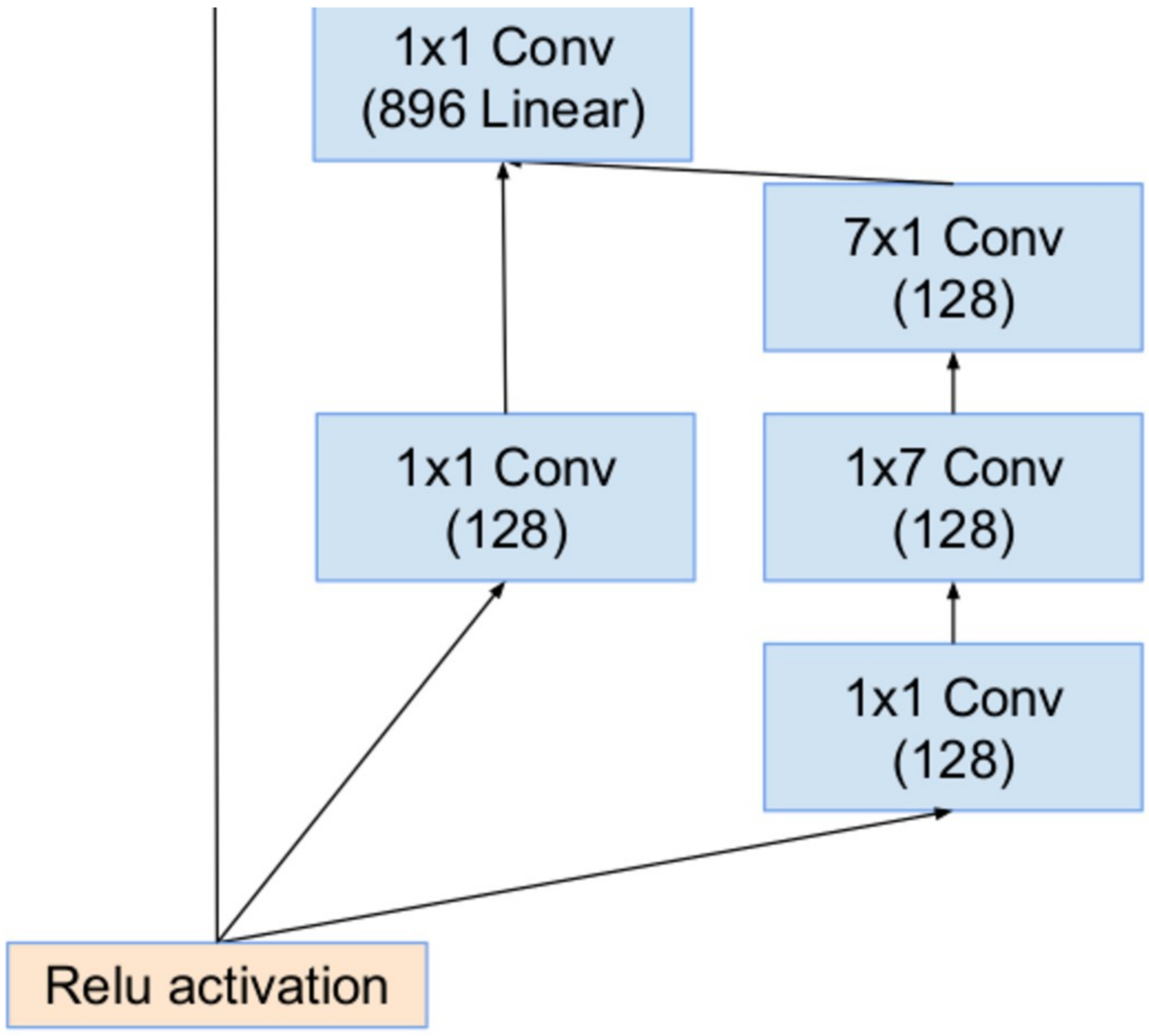


Figure 11. The schema for 17 × 17 grid (Inception-ResNet-B) module of Inception-ResNet-v1 network.


```python
# Inception-Renset-B
def block17(net, scale=1.0, activation_fn=tf.nn.relu, scope=None, reuse=None):
    """Builds the 17x17 resnet block."""
    with tf.variable_scope(scope, 'Block17', [net], reuse=reuse):
        with tf.variable_scope('Branch_0'):
            # 17*17*128
            tower_conv = slim.conv2d(net, 128, 1, scope='Conv2d_1x1')
        with tf.variable_scope('Branch_1'):
            # 17*17*128
            tower_conv1_0 = slim.conv2d(net, 128, 1, scope='Conv2d_0a_1x1')
            # 17*17*128
            tower_conv1_1 = slim.conv2d(tower_conv1_0, 128, [1, 7],
                                        scope='Conv2d_0b_1x7')
            # 17*17*128
            tower_conv1_2 = slim.conv2d(tower_conv1_1, 128, [7, 1],
                                        scope='Conv2d_0c_7x1')
        # 17*17*256
        mixed = tf.concat([tower_conv, tower_conv1_2], 3)
        # 17*17*896
        up = slim.conv2d(mixed, net.get_shape()[3], 1, normalizer_fn=None,activation_fn=None, scope='Conv2
        net += scale * up
        if activation_fn:
            net = activation_fn(net)
    return net

# 10 x Inception-Resnet-B
net = slim.repeat(net, 10, block17, scale=0.10)
end_points['Mixed_6b'] = net
```

Reduction-B

Reduction-B的输入为17*17*896，输出为8*8*1792。网络结构为：

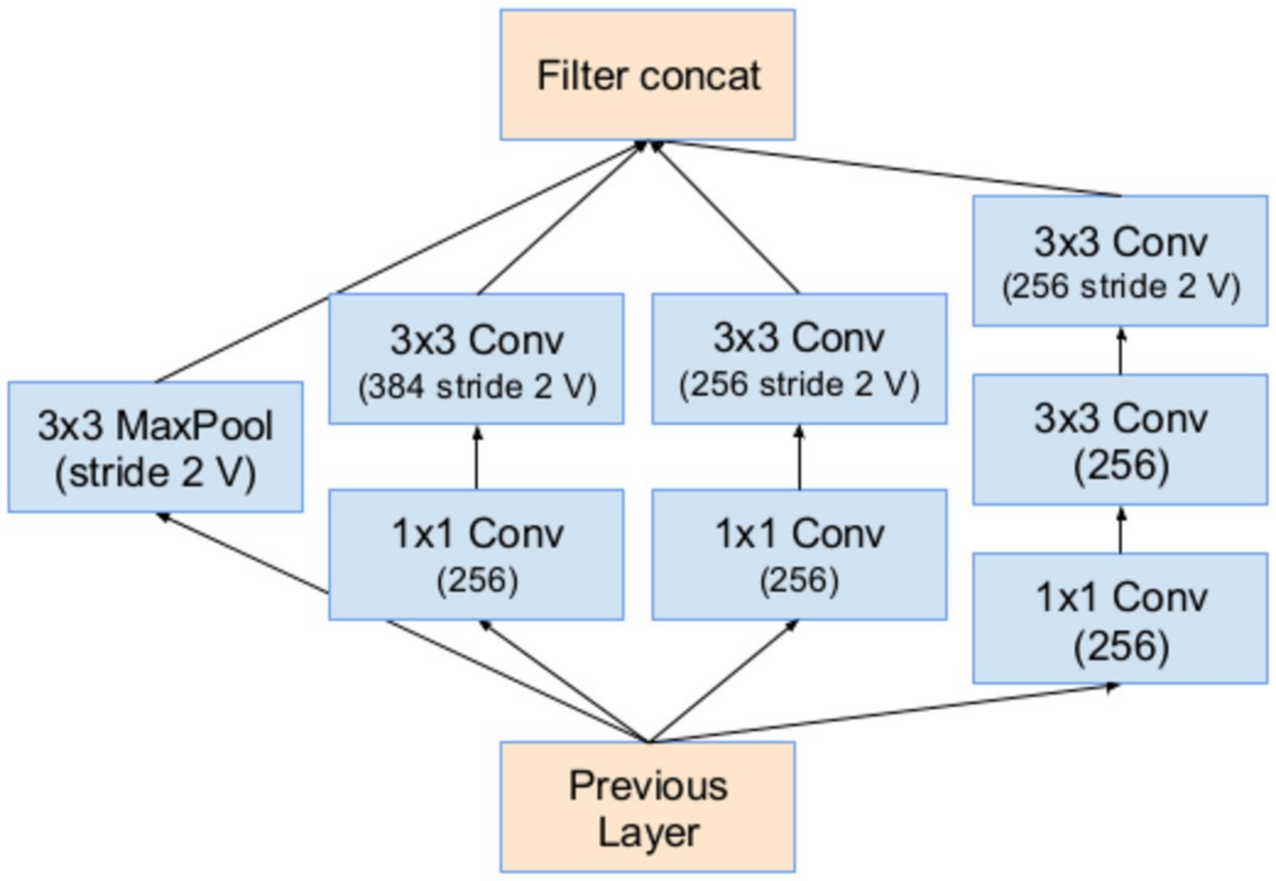


Figure 12. "Reduction-B" 17×17 to 8×8 grid-reduction module. This module used by the smaller Inception-ResNet-v1 network in Figure 15

对应的代码为：

```
def reduction_b(net):
    with tf.variable_scope('Branch_0'):
        # 17*17*256
        tower_conv = slim.conv2d(net, 256, 1, scope='Conv2d_0a_1x1')
        # 8*8*384
        tower_conv_1 = slim.conv2d(tower_conv, 384, 3, stride=2,
                                   padding='VALID', scope='Conv2d_1a_3x3')
    with tf.variable_scope('Branch_1'):
        # 17*17*256
        tower_conv1 = slim.conv2d(net, 256, 1, scope='Conv2d_0a_1x1')
        # 8*8*256
        tower_conv1_1 = slim.conv2d(tower_conv1, 256, 3, stride=2,
                                    padding='VALID', scope='Conv2d_1a_3x3')
    with tf.variable_scope('Branch_2'):
        # 17*17*256
        tower_conv2 = slim.conv2d(net, 256, 1, scope='Conv2d_0a_1x1')
        # 17*17*256
        tower_conv2_1 = slim.conv2d(tower_conv2, 256, 3,
                                    scope='Conv2d_0b_3x3')
        # 8*8*256
        tower_conv2_2 = slim.conv2d(tower_conv2_1, 256, 3, stride=2,
                                    padding='VALID', scope='Conv2d_1a_3x3')
    with tf.variable_scope('Branch_3'):
        # 8*8*896
        tower_pool = slim.max_pool2d(net, 3, stride=2, padding='VALID',
                                     scope='MaxPool_1a_3x3')
    # 8*8*1792
    net = tf.concat([tower_conv_1, tower_conv1_1,
                         tower_conv2_2, tower_pool], 3)
    return net
# Reduction-B
with tf.variable_scope('Mixed_7a'):
     net = reduction_b(net)
end_points['Mixed_7a'] = net
```

Inception-Resnet-C结构

Inception-Resnet-C结构重复5次。它输入为8*8*1792，输出为8*8*1792。对应的结构为：

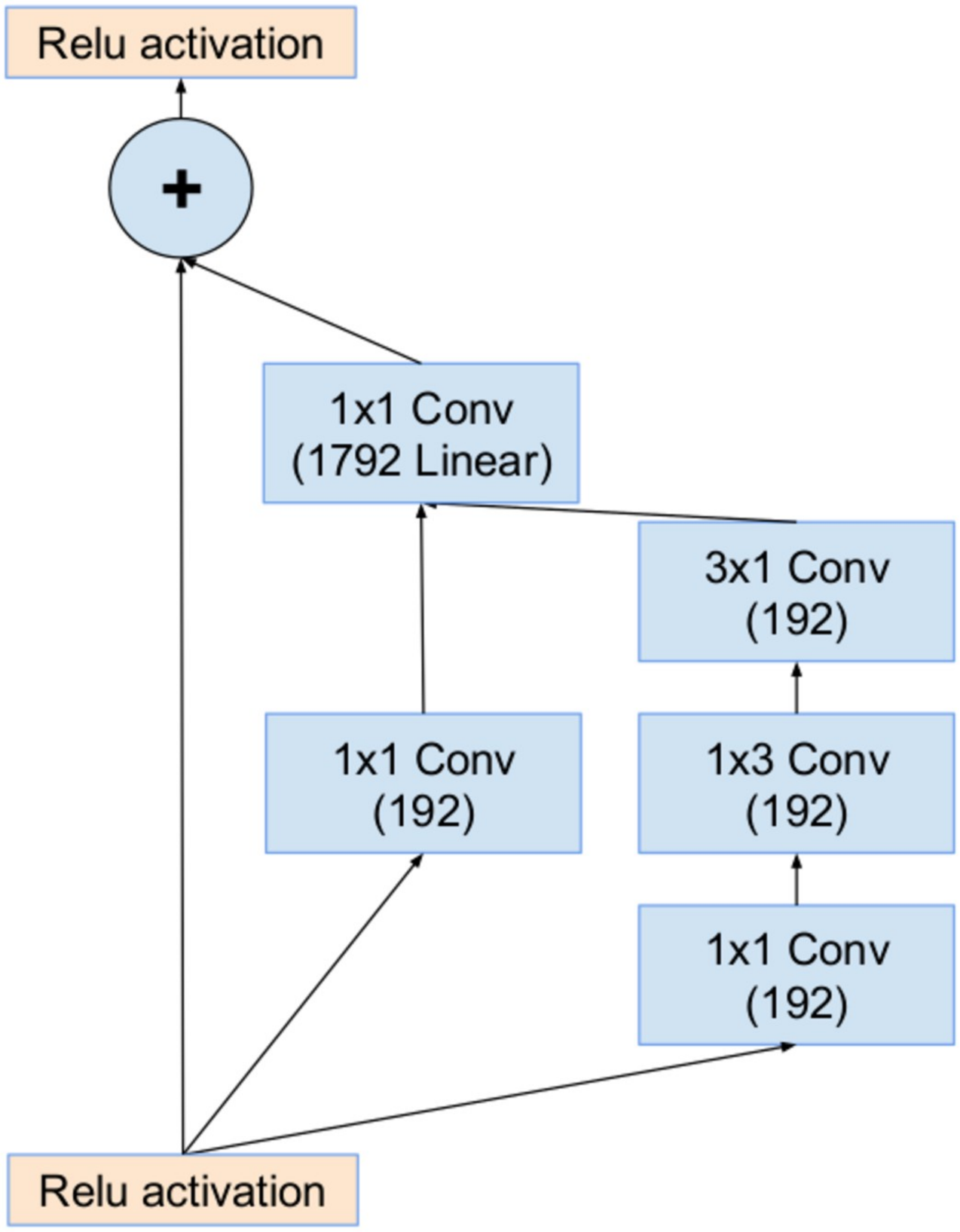


Figure 13. The schema for 8×8 grid (Inception-ResNet-C) module of Inception-ResNet-v1 network.

对应的代码为：

```python
# Inception-Resnet-C
def block8(net, scale=1.0, activation_fn=tf.nn.relu, scope=None, reuse=None):
    """Builds the 8x8 resnet block."""
    with tf.variable_scope(scope, 'Block8', [net], reuse=reuse):
        with tf.variable_scope('Branch_0'):
            # 8*8*192
            tower_conv = slim.conv2d(net, 192, 1, scope='Conv2d_1x1')
        with tf.variable_scope('Branch_1'):
            # 8*8*192
            tower_conv1_0 = slim.conv2d(net, 192, 1, scope='Conv2d_0a_1x1')
            # 8*8*192
            tower_conv1_1 = slim.conv2d(tower_conv1_0, 192, [1, 3],
                                        scope='Conv2d_0b_1x3')
            # 8*8*192
            tower_conv1_2 = slim.conv2d(tower_conv1_1, 192, [3, 1],
                                        scope='Conv2d_0c_3x1')
        # 8*8*384
        mixed = tf.concat([tower_conv, tower_conv1_2], 3)
        # 8*8*1792
        up = slim.conv2d(mixed, net.get_shape()[3], 1, normalizer_fn=None,activation_fn=None, scope='Conv2
        # scale=0.20
        net += scale * up
        if activation_fn:
            net = activation_fn(net)
    return net
# 5 x Inception-Resnet-C
net = slim.repeat(net, 5, block8, scale=0.20)
```

```
end_points['Mixed_8a'] = net
```

但是在facenet中，接下来又是一层Inception-Resnet-C，但是它没有重复，并且没有激活函数。输入与输出大小相同。

```
net = block8(net, activation_fn=None)
end_points['Mixed_8b'] = net
```

结果输出

结果输出包含Average Pooling和Dropout (keep 0.8)及Softmax三层，这里我们以facenet中为例：具体的代码如下：

```
with tf.variable_scope('Logits'):
    end_points['PrePool'] = net
    #pylint: disable=no-member
    # Average Pooling层，输出为8×8×1792
    net = slim.avg_pool2d(net, net.get_shape()[1:3], padding='VALID',scope='AvgPool_1a_8x8')
    #扁平除了batch_size维度的其它维度。使输出变为：[batch_size, ...]
    net = slim.flatten(net)
    #dropout层
    net = slim.dropout(net, dropout_keep_prob, is_training=is_training,scope='Dropout')
    end_points['PreLogitsFlatten'] = net
    # 全链接层。输出为batch_size×128
    net = slim.fully_connected(net, bottleneck_layer_size, activation_fn=None,scope='Bottleneck', reuse=F
```