NIN Notes

1. Architecture

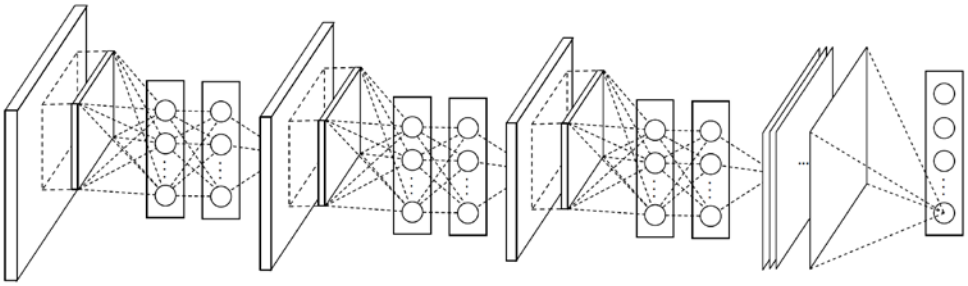

Figure 2: The overall structure of Network In Network. In this paper the NINs include the stacking of three mlpconv layers and one global average pooling layer.

- 一般来说，如果要提取的一些潜在的特征是线性可分的话，那么对于线性的卷积运算来说就足够了。然而我们所要提取的特征一般是高度非线性的。在传统的 CNN 中，也许我们可以用超完备的滤波器，来提取各种潜在的特征。比如我们要提取某个特征，于是我就用了一大堆的滤波器，把所有可能的提取出来，这样就可以把我想要提取的特征也覆盖到，然而这样存在一个缺点，那就是网络太恐怖了，参数太多了。
- 我们知道 CNN 高层特征其实是低层特征通过某种运算的组合。于是作者就根据这个想法，提出在每个局部感受野中进行更加复杂的运算，提出了对卷积层的改进算法：MLP 卷积层。另一方面，传统的 CNN 最后一层都是全连接层，参数个数非常之多，容易引起过拟合，一个 CNN 模型，大部分的参数都被全连接层给占用了，故这篇 paper 采用了全局均值池化，替代全连接层。

2. MLP 卷积层

- 本文中用到的网络结构是三个 MLP 卷积层+一个全局平均池化层，可以从上图中看出来。
- 下面代码是一个 MLP 卷积层的定义

```
# conv1 layer
with tf.name_scope('conv1'):
    W_conv1 = weight_variable([5, 5, 3, 192], stddev=0.01)
    b_conv1 = tf.Variable(tf.random_normal(
        [192], stddev=0.01, type=tf.float32))
    output = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
    print('conv1 output: ', output.shape)   # [None, 32, 32, 3]
    tf.summary.histogram('conv_filter', output)
    tf.summary.scalar('conv_filter', tf.nn.zero_fraction(output))
    # MLP-1-1
    with tf.name_scope('mlp_1_1'):
        W_MLP11 = weight_variable([1, 1, 192, 160])
        b_MLP11 = bias_variable([160])
        output = tf.nn.relu(conv2d(output, W_MLP11) + b_MLP11)
```

```
        print('MLP-1-1 output: ', output.shape)     # [None, 32, 32, 160]
        tf.summary.histogram('mlp', output)
        tf.summary.scalar('mlp', tf.nn.zero_fraction(output))
    # MLP-1-2
    with tf.name_scope('mlp_1_2'):
        W_MLP12 = weight_variable([1, 1, 160, 96])
        b_MLP12 = bias_variable([96])
        output = tf.nn.relu(conv2d(output, W_MLP12) + b_MLP12)
        print('MLP-1-2 output: ', output.shape)     # [None, 32, 32, 96]
        tf.summary.histogram('mlp', output)
        tf.summary.scalar('mlp', tf.nn.zero_fraction(output))

    with tf.variable_scope('Visualization'):
        grid = put_kernels_on_grid(W_conv1)
        tf.summary.image('conv1/filters', grid, max_outputs=1)

    # Max pooling
    output = max_pool_3x3(output)
    # dropout
    output = tf.nn.dropout(output, keep_prob)
```

3. Global Average Pooling

- 代码如下

```
# global average
output = tf.nn.avg_pool(output, ksize=[1, 8, 8, 1], strides=[1, 1, 1, 1], padding='VALID')
```