GoogLeNet

Going deeper with convolutions

이민주

1. Introduction

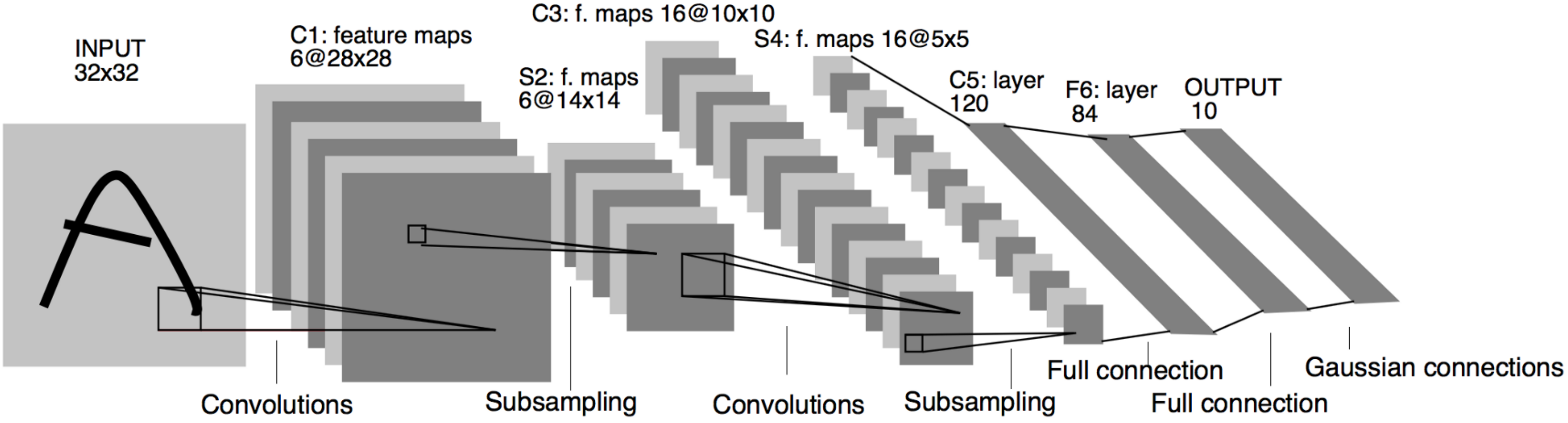
- Increasing the depth and width of the network while keeping the computational budget constant

  - 12X fewer parameters, more accurate
  - Efficient deep neural network architecture

2. Related Work

- LeNet-5 : standard structure

➔ Stacked conv layers, one or more fully-connected layers

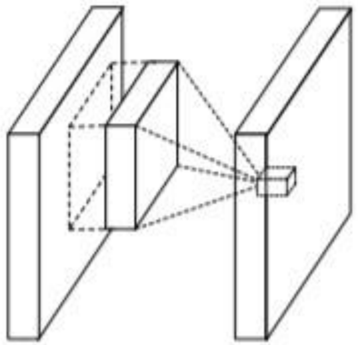


2. Related Work

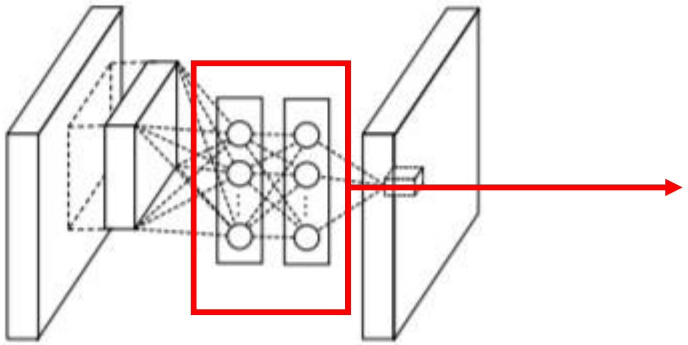
- Network-in-Network

→ Additional 1X1 conv layers

→ Dimension reduction modules to remove computational bottlenecks, limit the size of networks
→ Increasing the depth and width of networks without performance penalty



(a) Linear convolution layer



(b) Mlpconv layer

MLP (Multi Layer Perceptron)

$$f^{1}_{i,j,k_1} = max(w^{1\top}_{k_1} x_{i,j} + b_{k_1}, 0)$$

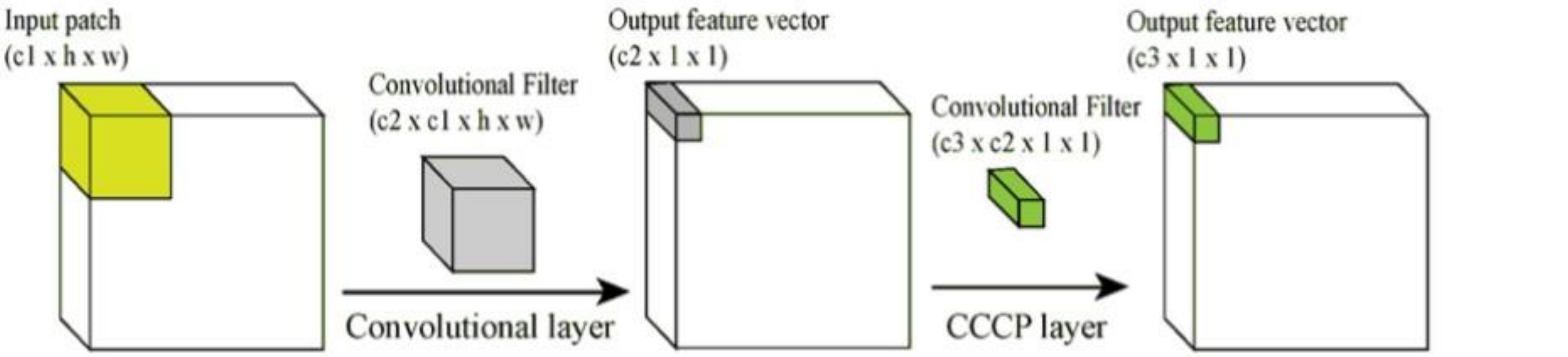
$$f^{n}_{i,j,k_n} = max(w^{n\top}_{k_n} f^{n-1}_{i,j} + b_{k_n}, 0)$$

2. Related Work

- Network-in-Network

→ Additional 1X1 conv layers

→ Dimension reduction modules to remove computational bottlenecks, limit the size of networks
→ Increasing the depth and width of networks without performance penalty



Efficient implementation of CCCP

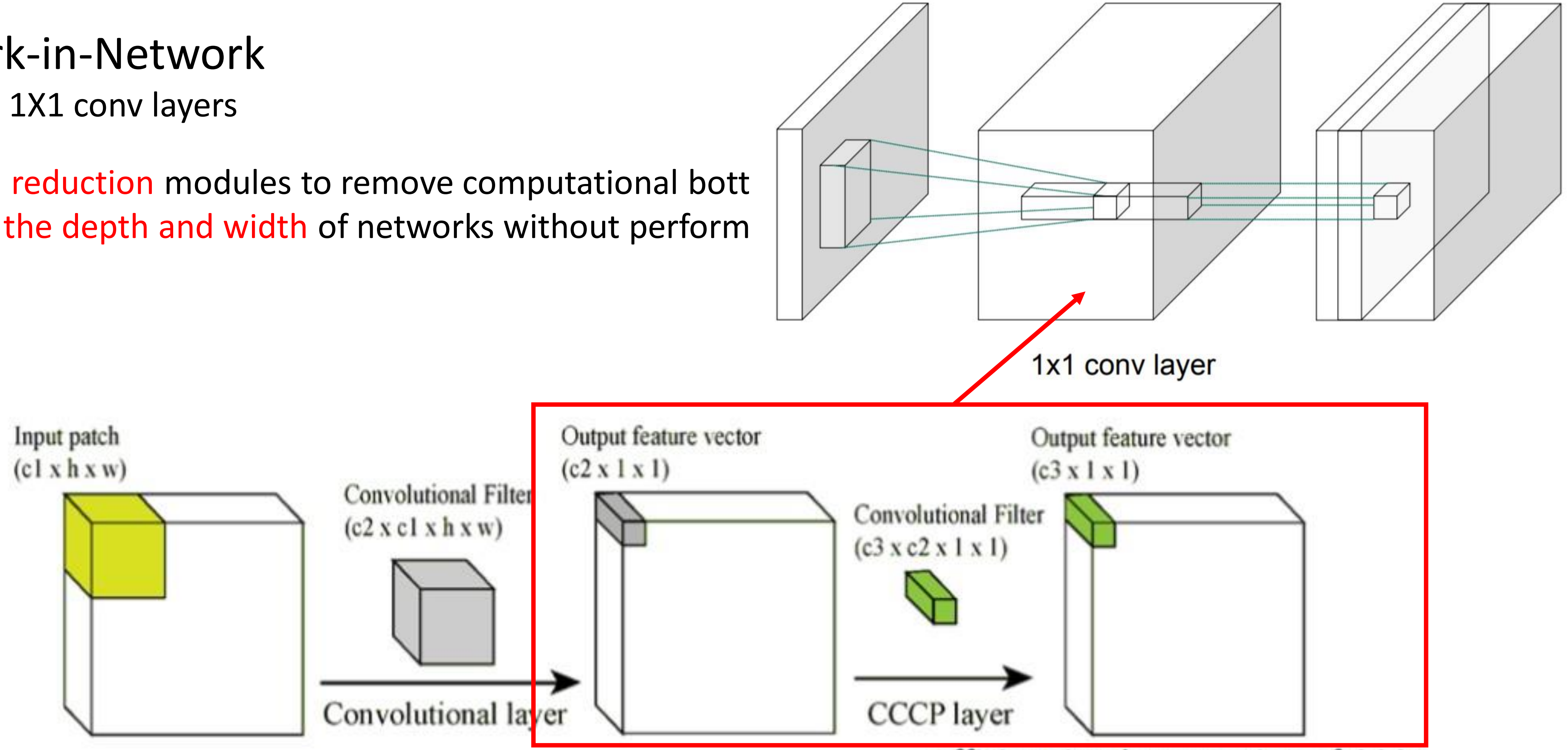
2. Related Work

- Network-in-Network

→ Additional 1X1 conv layers

→ Dimension reduction modules to remove computational bott

→ Increasing the depth and width of networks without perform



Efficient implementation of CCCP

3. Motivation and High Level Considerations

- Improving the performance of DNN

→ Increasing the depth(the number of levels) of the network
→ Increasing the width(the number of units at each level)

P1 . Larger number of parameters, overfitting
P2. uniformly increased network size is the increased use of computational resources
 If the added capacity is used inefficiently, the a lot of computation is wasted

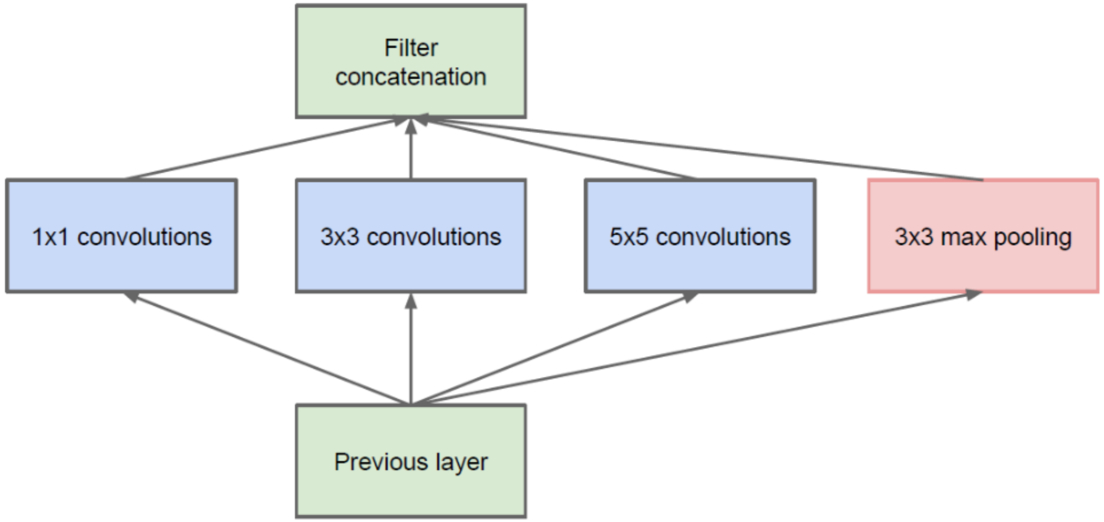
S1. ultimately moving from fully connected to sparsely connected architectures

4. Architectural Details

- Inception architecture

➔ Finding out how an optimal local sparse structure can be approximated and covered by readily available dense components

→ The optimal local construction & repeat spatially
→ the correlation statistics of the last layer and cluster them into groups of units with high correlation

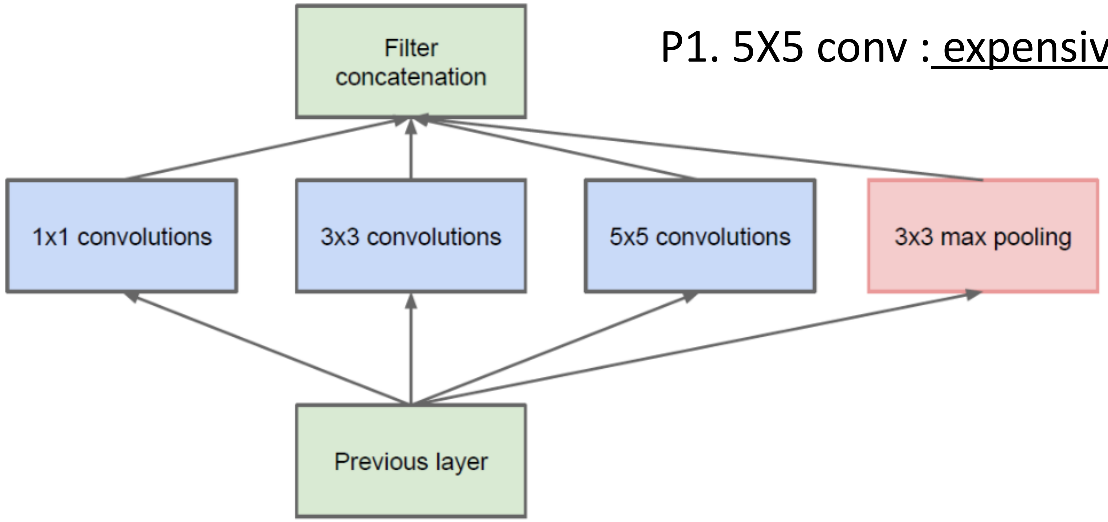


(a) Inception module, naïve version

4. Architectural Details

- 1X1 conv : a lot of clusters concentrated in a single region
- 3X3 & 5X5 conv : be covered by convolutions over larger patches,
  decreasing number of patches over larger and larger regions.
- Pooling : additional beneficial effect

➔ Combination of layers , concentrated into a single output vector forming the input of the next stage



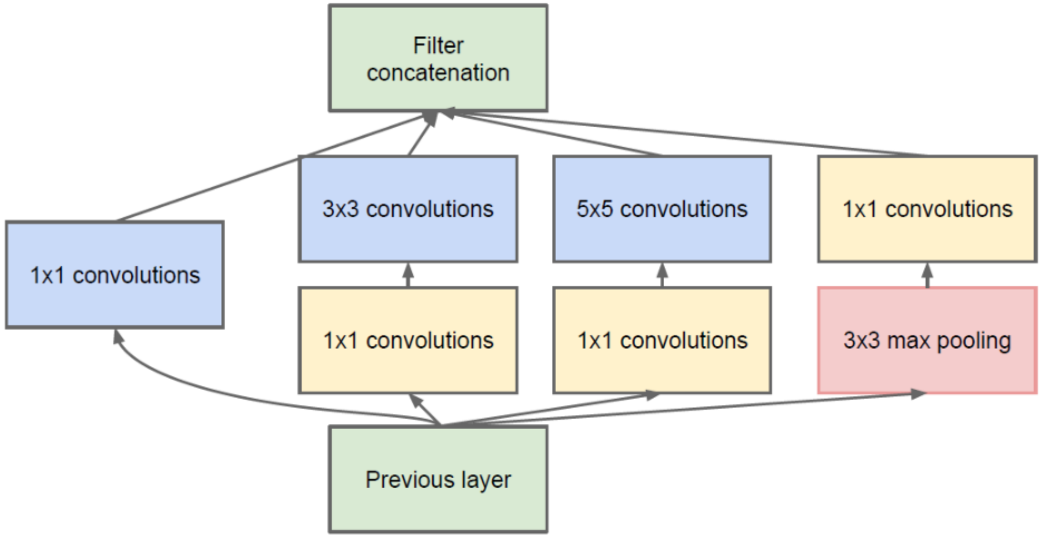
(a) Inception module, naïve version

P1. 5X5 conv : expensive with a large number of filters

4. Architectural Details

- Applying dimension reductions and projections
- Low dimensional embeddings contain a lot of information & keep representation sparse

➔ 1X1 conv : compute reductions before the expensive 3X3 and 5X5 convs
Reduction & include the use of rectified linear activation



(b) Inception module with dimension reductions

4. Architectural Details

```
inception_3a_1x1 = Convolution2D(64,1,1,border_mode='same',activation='relu',name='inception_3a/1x1',W_regularizer=l2(0.0002))(pool2_3x3_s2)

inception_3a_3x3_reduce = Convolution2D(96,1,1,border_mode='same',activation='relu',name='inception_3a/3x3_reduce',W_regularizer=l2(0.0002))(pool2_3x3_s2)

inception_3a_3x3 = Convolution2D(128,3,3,border_mode='same',activation='relu',name='inception_3a/3x3',W_regularizer=l2(0.0002))(inception_3a_3x3_reduce)

inception_3a_5x5_reduce = Convolution2D(16,1,1,border_mode='same',activation='relu',name='inception_3a/5x5_reduce',W_regularizer=l2(0.0002))(pool2_3x3_s2)

inception_3a_5x5 = Convolution2D(32,5,5,border_mode='same',activation='relu',name='inception_3a/5x5',W_regularizer=l2(0.0002))(inception_3a_5x5_reduce)

inception_3a_pool = MaxPooling2D(pool_size=(3,3),strides=(1,1),border_mode='same',name='inception_3a/pool')(pool2_3x3_s2)

inception_3a_pool_proj = Convolution2D(32,1,1,border_mode='same',activation='relu',name='inception_3a/pool_proj',W_regularizer=l2(0.0002))(inception_3a_pool)

inception_3a_output = merge([inception_3a_1x1,inception_3a_3x3,inception_3a_5x5,inception_3a_pool_proj],mode='concat',concat_axis=1,name='inception_3a/output')


inception_3b_1x1 = Convolution2D(128,1,1,border_mode='same',activation='relu',name='inception_3b/1x1',W_regularizer=l2(0.0002))(inception_3a_output)

inception_3b_3x3_reduce = Convolution2D(128,1,1,border_mode='same',activation='relu',name='inception_3b/3x3_reduce',W_regularizer=l2(0.0002))(inception_3a_output)

inception_3b_3x3 = Convolution2D(192,3,3,border_mode='same',activation='relu',name='inception_3b/3x3',W_regularizer=l2(0.0002))(inception_3b_3x3_reduce)

inception_3b_5x5_reduce = Convolution2D(32,1,1,border_mode='same',activation='relu',name='inception_3b/5x5_reduce',W_regularizer=l2(0.0002))(inception_3a_output)

inception_3b_5x5 = Convolution2D(96,5,5,border_mode='same',activation='relu',name='inception_3b/5x5',W_regularizer=l2(0.0002))(inception_3b_5x5_reduce)

inception_3b_pool = MaxPooling2D(pool_size=(3,3),strides=(1,1),border_mode='same',name='inception_3b/pool')(inception_3a_output)

inception_3b_pool_proj = Convolution2D(64,1,1,border_mode='same',activation='relu',name='inception_3b/pool_proj',W_regularizer=l2(0.0002))(inception_3b_pool)

inception_3b_output = merge([inception_3b_1x1,inception_3b_3x3,inception_3b_5x5,inception_3b_pool_proj],mode='concat',concat_axis=1,name='inception_3b/output')


inception_3b_output_zero_pad = ZeroPadding2D(padding=(1, 1))(inception_3b_output)

pool3_helper = PoolHelper()(inception_3b_output_zero_pad)

pool3_3x3_s2 = MaxPooling2D(pool_size=(3,3),strides=(2,2),border_mode='valid',name='pool3/3x3_s2')(pool3_helper)
```

5. GoogLeNet

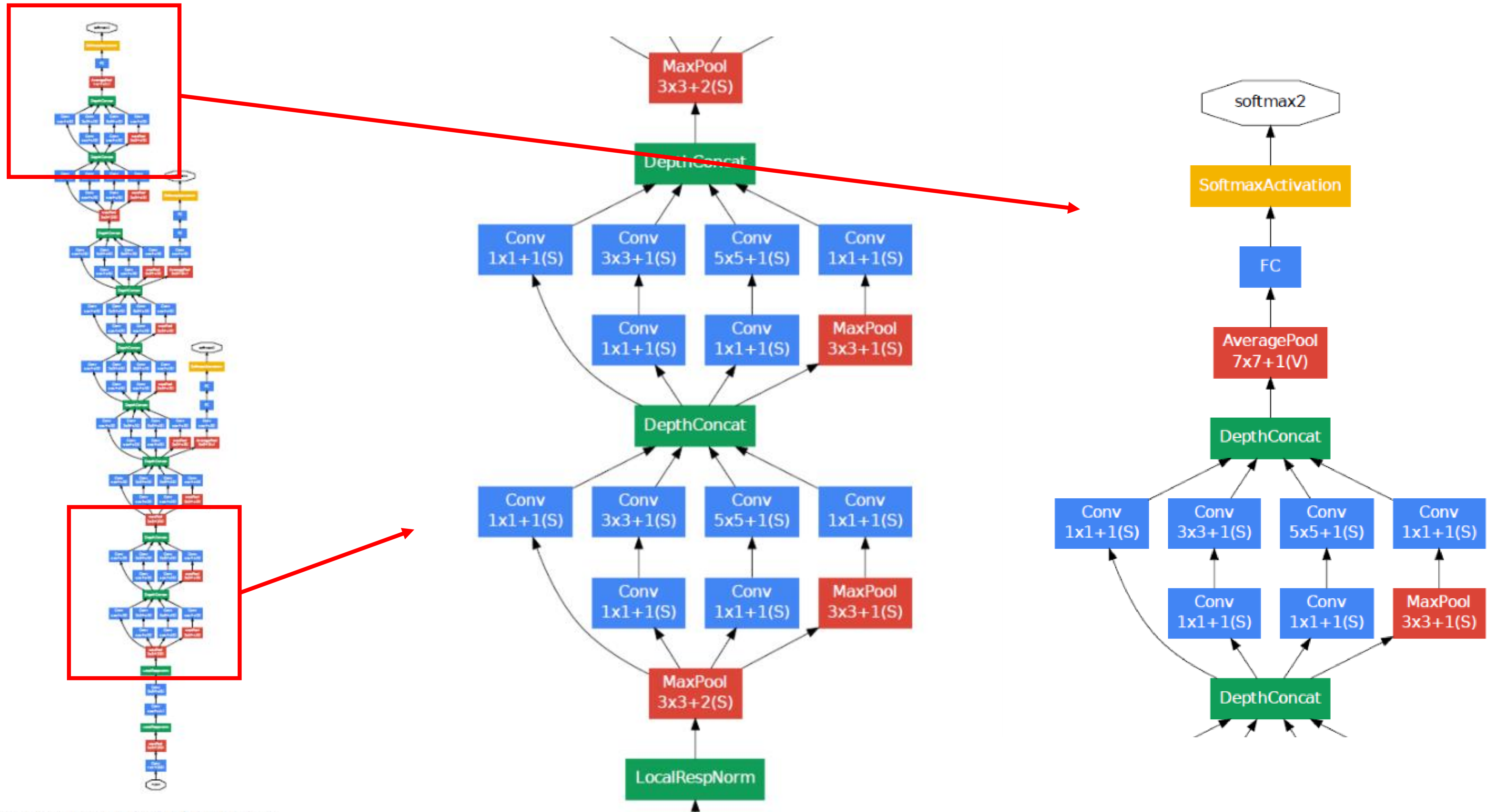


Figure 3: GoogLeNet network with all the bells and whistles

5. GoogLeNet

- # 3X3 reduce : number of 1X1 filters before 3X3 conv …

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

5. GoogLeNet

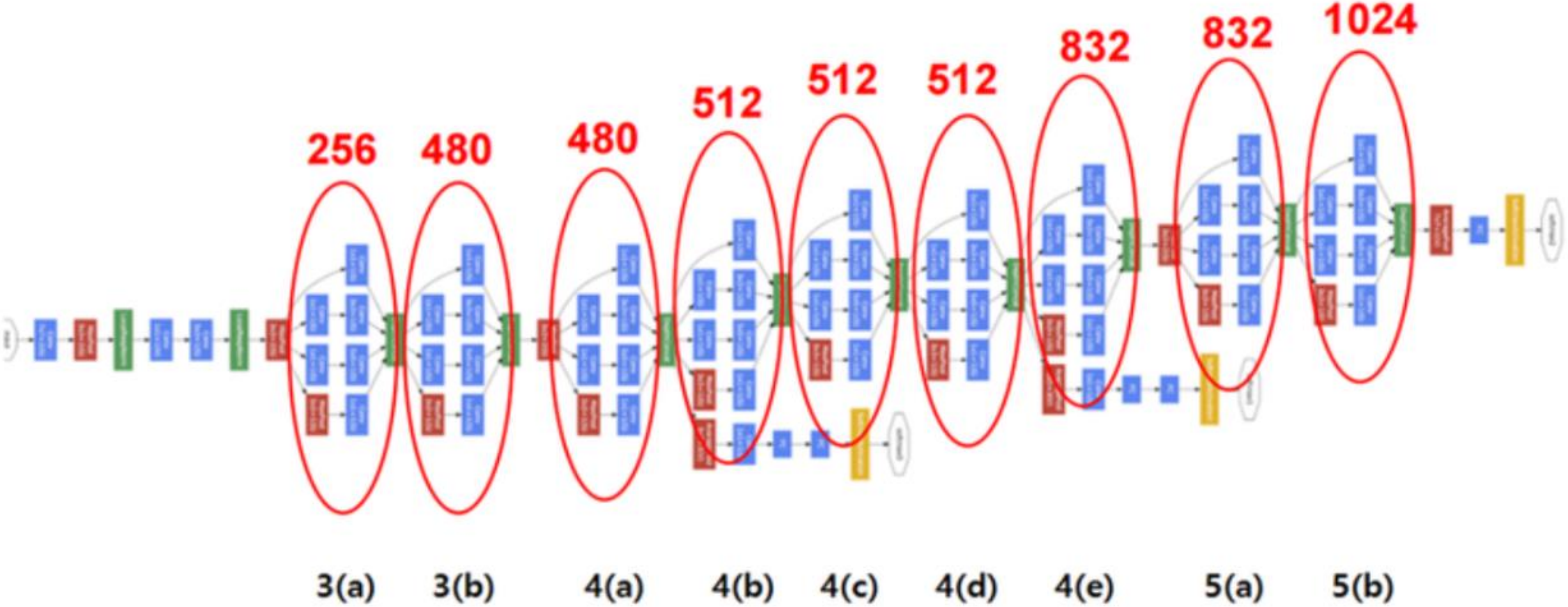


Table 1: GoogLeNet incarnation of the Inception architecture

5. GoogLeNet

- 22 layers deep
- Average pooling before classifier
  → average pooling은 spatial한 정보를 합하는 방식이기 때문에 입력 이미지의 spatial 변환에 robust 하다
  → fully connected layer보다 0.6% 향상 (dropout 추가)
- By adding auxiliary classifiers on top of 4a and 4b, increase backprop gradient
  → loss added to the total loss , weighted by 0.3
  → average pooling 5X5 filter, stride 3