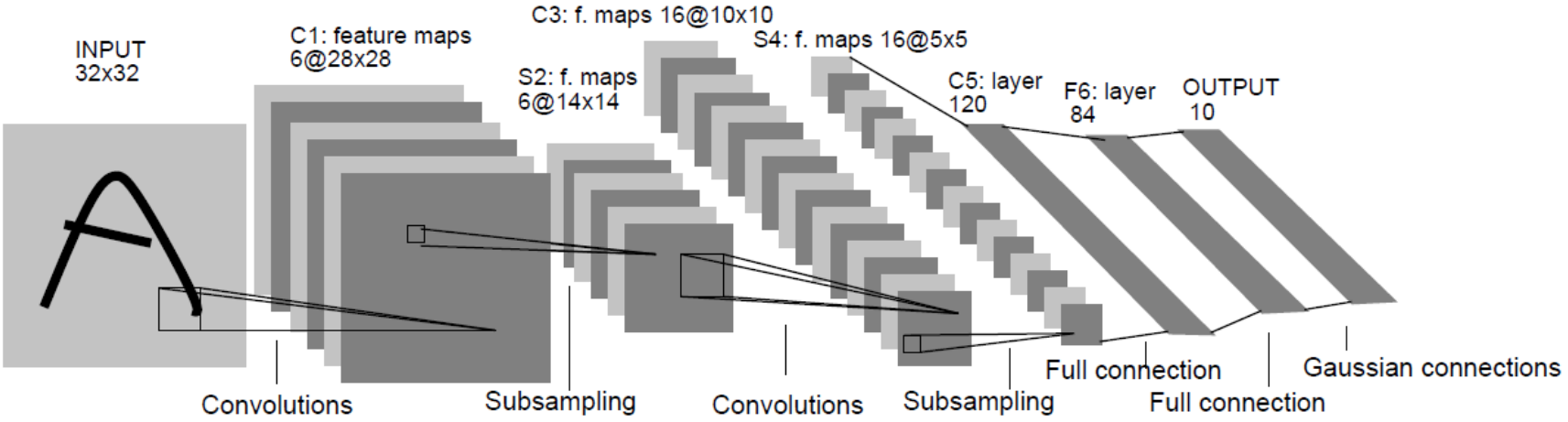
Caffe Parameters

Lenet5



C1

Input

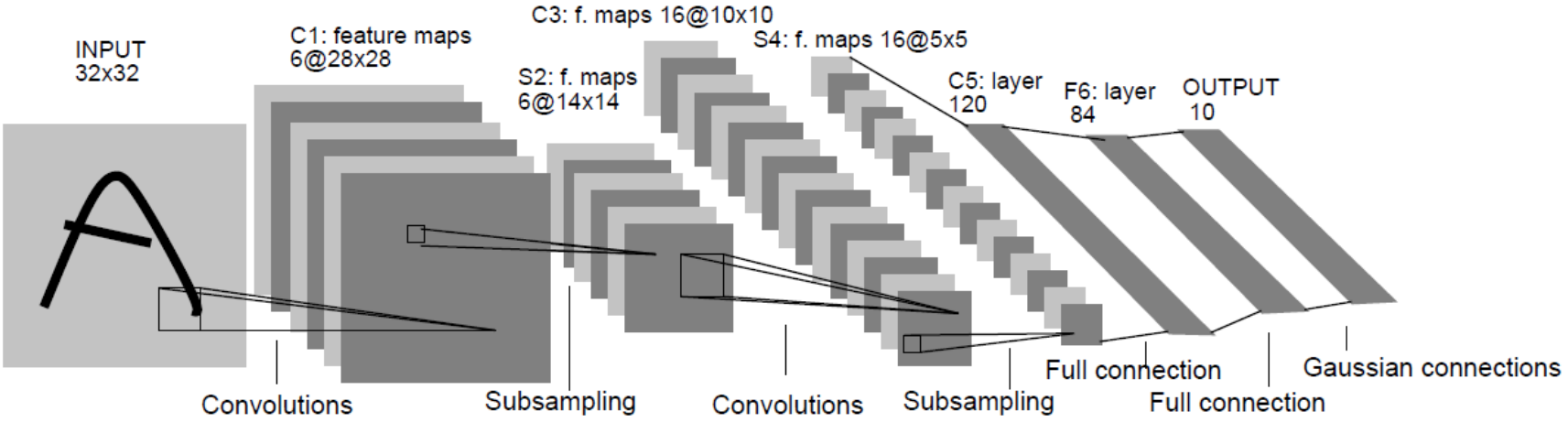
1@32x32

Output

6@28x28

Kernel : 6@5x5x1
Params : 6@5x5x1 : (5x5)x (1x6) +6 = 120

Lenet5



C3

Input **Output**

6@14x14 **16@10x10**

Kernel : 16@10x10x6

Params 16@5x5x6 : (5x5)x (6 x16) +16 = ~~2416~~

Params 16@5x5x6 : (5x5) x(3x6 + 4*9 + 6*1) + 16 = 1,516

Output 16ch

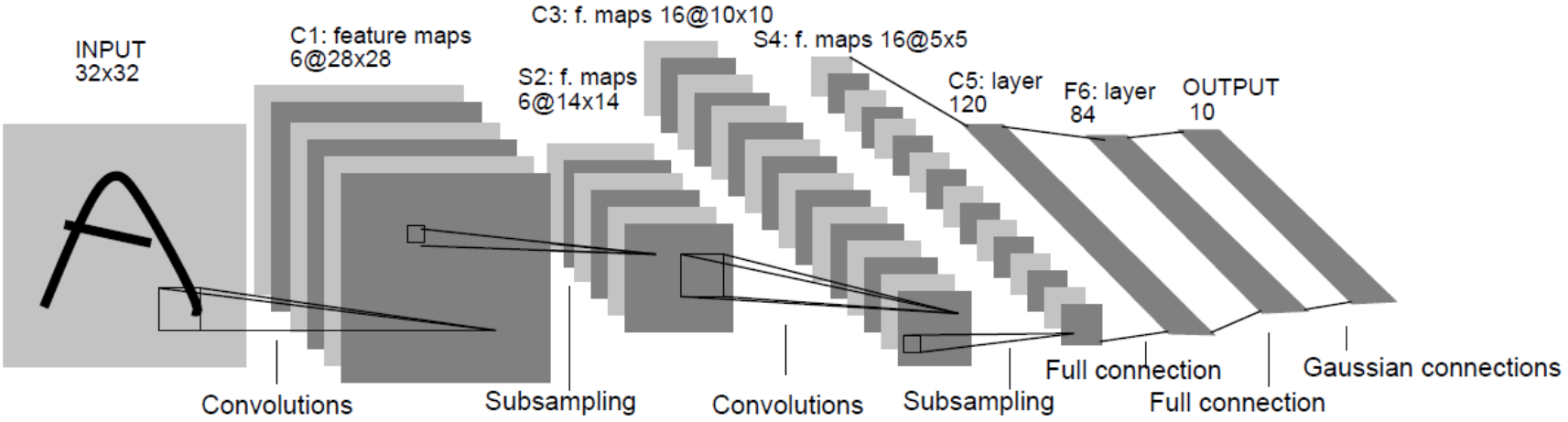
Input 6ch

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

TABLE I

EACH COLUMN INDICATES WHICH FEATURE MAP IN S2 ARE COMBINED BY THE UNITS IN A PARTICULAR FEATURE MAP OF C3.

Lenet5



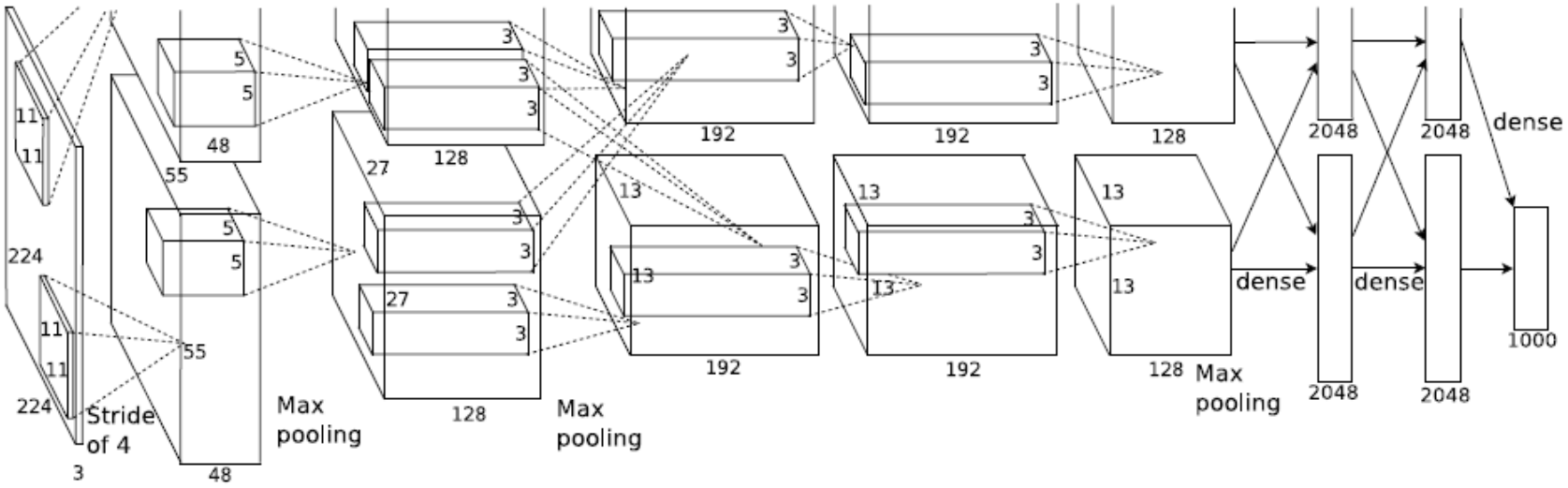
C5

Input **Output**

16@5x5 **120@1x1**

Kernel : 120@5x5x16
Params : 120@5x5x16 : (5x5)x (16 x120) +120 = 48,120

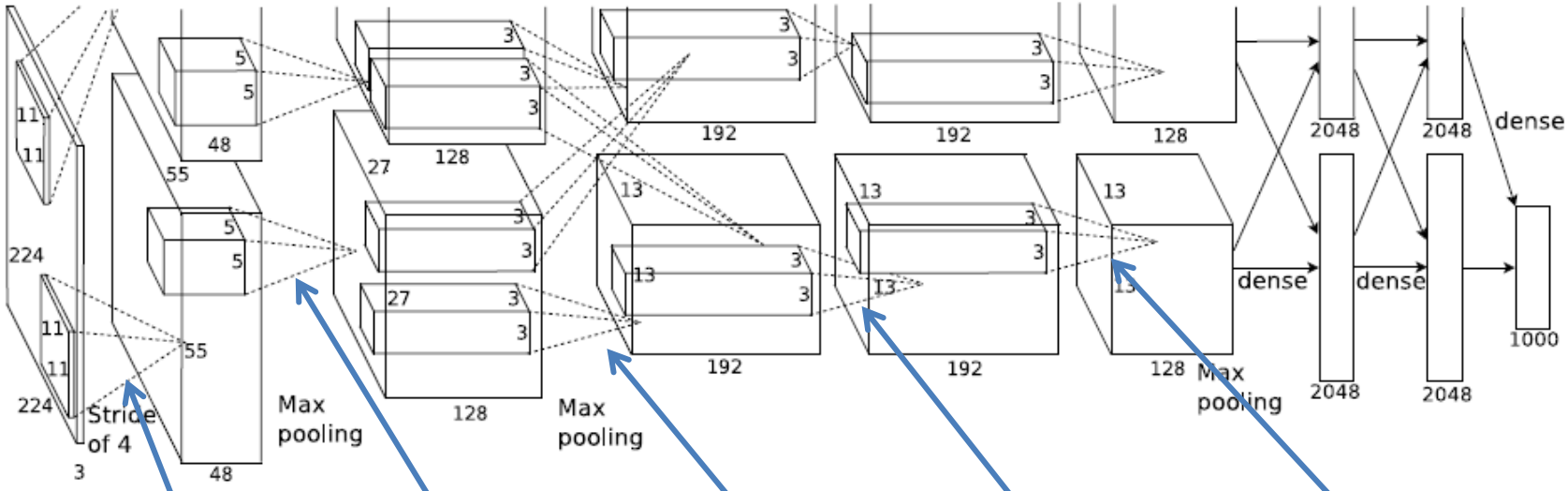
Alexnet(original)



Alexnet(flat)

Input

3@	96@	256@	384@	384@	256@	4096	4096	1000
224x224	55x55	27x27	13x13	13x13	13x13			
150528	290400	186624	64896	64896	43264			



Kernel

C1 96@11x11x <u>3</u>

C2 256@5x5x <u>48</u> half

C3 384@3x3x <u>256</u> full

C4 384@3x3x <u>192</u> half

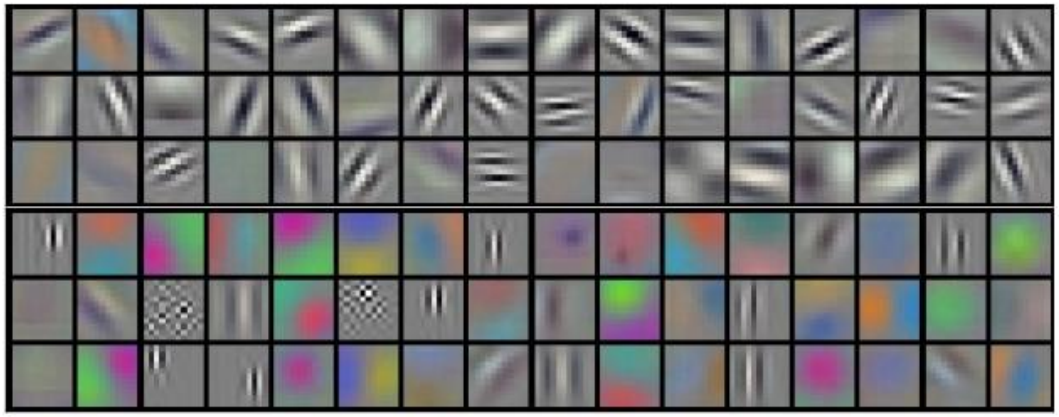
C5 256@3x3x <u>192</u> half

Params in Convolution Filters

96 @ 11x11x 3
256 @ 5x 5 x 48
384 @ 3x 3 x 256
384 @ 3x 3 x 192
256 @ 3x 3 x 192

96 x 3 @ 11x11
256 x 48 @ 5x 5
384 x 256 @ 3x 3
384 x 192 @ 3x 3
256 x 192 @ 3x 3

96x3@11x11



```
model_def = param_root + 'deploy.prototxt'
model_weights = param_root + 'bvlc_reference_caffenet.caffemodel‘

net = caffe.Net(model_def,      # defines the structure of the model
                model_weights,  # contains the trained weights
                caffe.TEST)     # use test mode (e.g., don't perform dropout)

for layer_name, blob in net.blobs.iteritems():
    print layer_name + '\t' + str(blob.data.shape)

for layer_name, param in net.params.iteritems():
    print layer_name + '\t' + str(param[0].data.shape), str(param[1].data.shape)
```

```
data        (50, 3, 227, 227)
conv1       (10, 96, 55, 55)
pool1       (10, 96, 27, 27)
norm1       (10, 96, 27, 27)
conv2       (10, 256, 27, 27)
pool2       (10, 256, 13, 13)
norm2       (10, 256, 13, 13)
conv3       (10, 384, 13, 13)
conv4       (10, 384, 13, 13)
conv5       (10, 256, 13, 13)
pool5       (10, 256, 6, 6)
fc6         (10, 4096)
fc7         (10, 4096)
fc8         (10, 1000)
prob        (10, 1000)
```

```
conv1       (96, 3, 11, 11) (96,)
conv2       (256, 48, 5, 5) (256,)
conv3       (384, 256, 3, 3) (384,)
conv4       (384, 192, 3, 3) (384,)
conv5       (256, 192, 3, 3) (256,)
fc6         (4096, 9216) (4096,)
fc7         (4096, 4096) (4096,)
fc8         (1000, 4096) (1000,)
```

vis_square

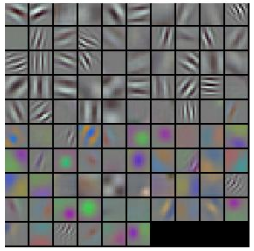
```
def vis_square(data, padsize=1, padval=0):
    data -= data.min()
    data /= data.max()

    # force the number of filters to be square
    n = int(np.ceil(np.sqrt(data.shape[0])))
    padding = ((0, n ** 2 - data.shape[0]), (0, padsize), (0, padsize)) + ((0, 0),) * (data.ndim - 3)
    data = np.pad(data, padding, mode='constant', constant_values=(padval, padval))

    # tile the filters into an image
    data = data.reshape((n, n) + data.shape[1:]).transpose((0, 2, 1, 3) + tuple(range(4, data.ndim + 1)))
    data = data.reshape((n * data.shape[1], n * data.shape[3]) + data.shape[4:])

    plt.imshow(data) ; plt.axis('off')
    plt.show()
```

```
filters = net.params['conv1'][0].data
vis_square(filters.transpose(0, 2, 3, 1))

filters = net.params['conv2'][0].data
vis_square(filters[:48].reshape(48**2, 5, 5))

filters = net.params['conv3'][0].data
vis_square(filters[:256].reshape(256**2, 3, 3))

ilters = net.params['conv4'][0].data
vis_square(filters[:192].reshape(192**2, 3, 3))

ilters = net.params['conv5'][0].data
vis_square(filters[:192].reshape(192**2, 3, 3))
```

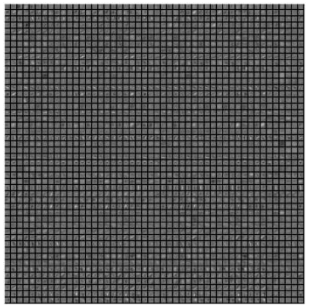
Pre-trained RAW parameters

Conv1

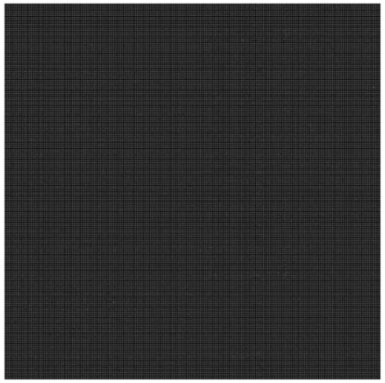
11x11



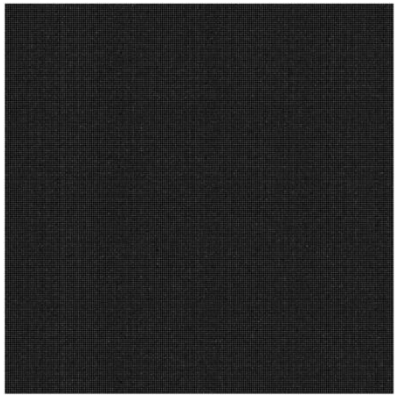
Conv2



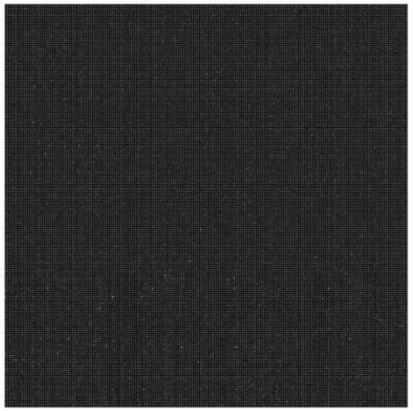
Conv3



Conv4



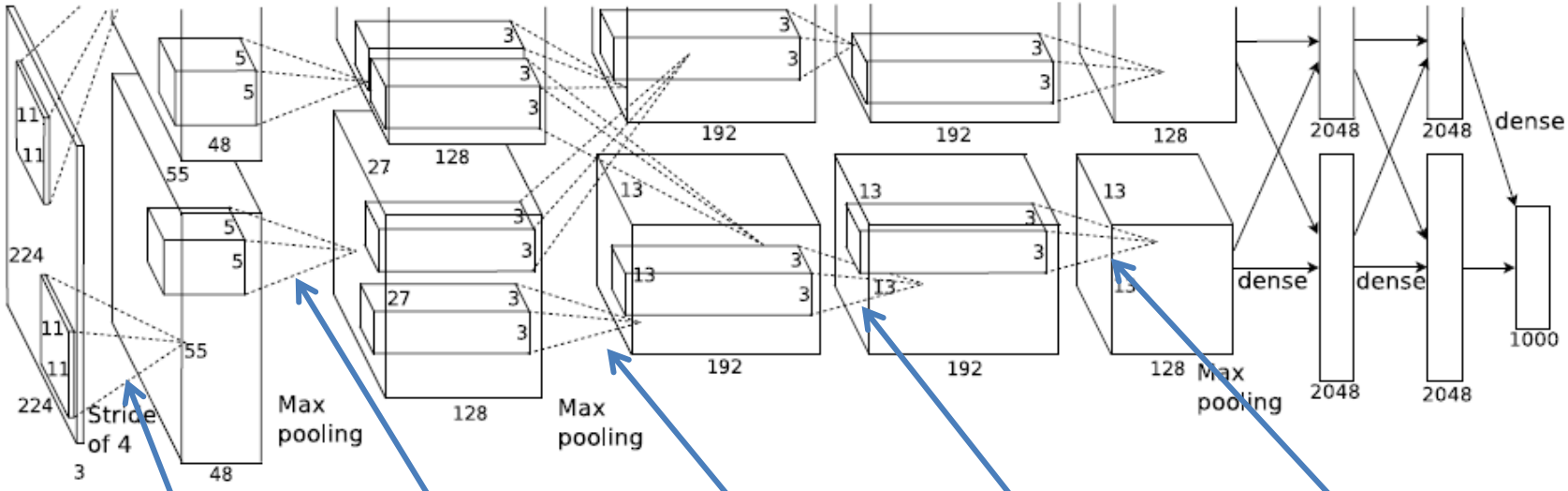
Conv5



Alexnet(OWT)

Input

3@	64@	192@	384@	384@	256@	4096	4096	1000
224x224	55x55	27x27	13x13	13x13	13x13			
150528	193600	1399688	64896	64896	43264			



Kernel

C1 64@11x11x 3

C2 192@5x5x 64

C3 384@3x3x 192

C4 384@3x3x 384

C5 256@3x3x 384