50.035 2018 Mid-term

Sol. for selected questions

[20 points] Question 2: Inception module and GoogLeNet

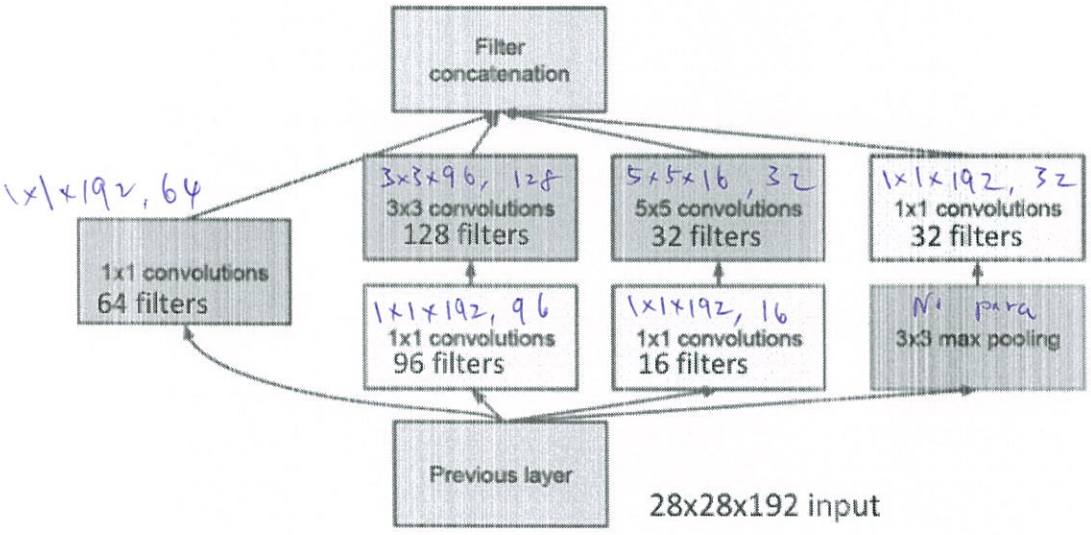
What is the purpose of using 1x1 convolutions before 5x5 and 3x3 convolutions in the Inception module?

see "class_09" notes

Calculate the number of trainable parameters in the following inception module. There is no need to include the bias of the filter. **Show your calculation.**

What is the number of channels after "Filter concatenation"?

Note:
1K = 1024
1M = 1024 * 1024
Recall that in an Inception module, all convolution and max pooling outputs have the same spatial dimension as their inputs. Moreover, max pooling uses stride=1.

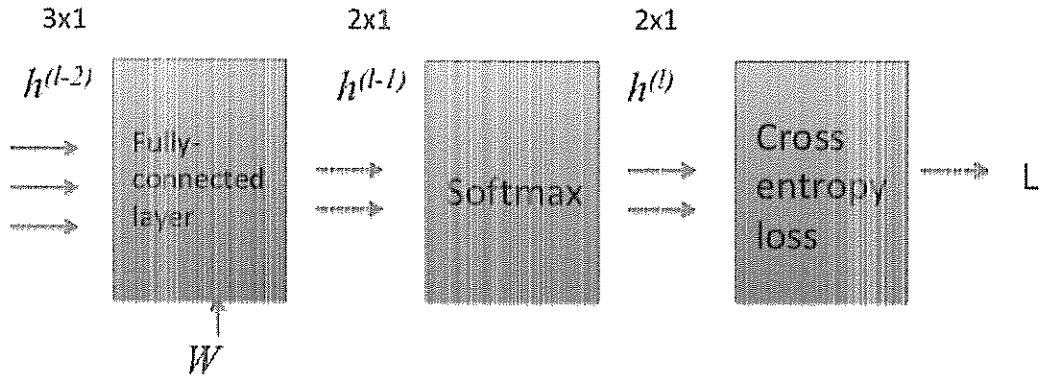


trainable parameters = $1\times1\times192\times64 + 1\times1\times192\times96 + 1\times1\times192\times16$
$3\times3\times96\times128 + 5\times5\times16\times32 + 1\times1\times192\times32$
$= 159.5k$

channels after filter concatenation $= 64 + 128 + 32 + 32$
$= 256$

[20 points]
c) The figure below shows the last several layers of a deep neural network trained with cross-entropy loss L.

Note that the first class is the ground-truth class for this training sample.



$$W = \begin{bmatrix} 5 & 4 & 3 \\ 3 & 4 & 2 \end{bmatrix}$$

Given the 3-by-1 activation vector: $h^{(l-2)} = [0.3,0.2,0.2]^T$ as input to the fully-connected layer, compute: (show your steps)

(i) L

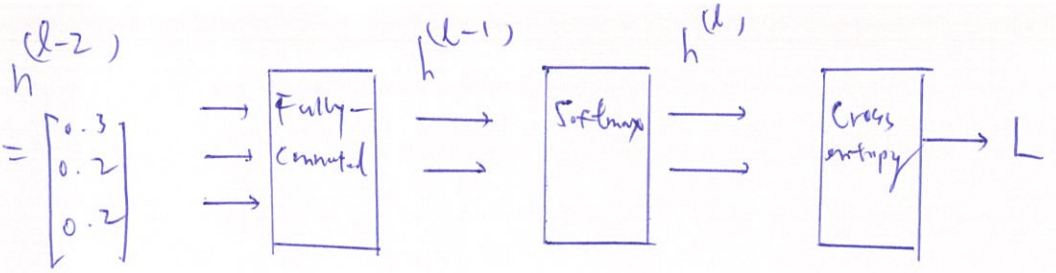
(ii) $\dfrac{\partial L}{\partial W}$

(iii) $\dfrac{\partial L}{\partial h^{(l-2)}}$

(iv) The new W updated with gradient descent with a step size:

$$\gamma = 2$$

3(c)



$$W = \begin{bmatrix} 5 & 4 & 3 \\ 3 & 4 & 2 \end{bmatrix}$$

(i) L:

$$h^{(l-1)} = W h^{(l-2)} = \begin{bmatrix} 5 & 4 & 3 \\ 3 & 4 & 2 \end{bmatrix} \begin{bmatrix} 0.3 \\ 0.2 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 2.9 \\ 2.1 \end{bmatrix}$$

$$h^{(l)} = \text{softmax}\left(h^{(l-1)}\right) = \begin{bmatrix} \frac{e^{2.9}}{e^{2.9}+e^{2.1}} \\ \frac{e^{2.1}}{e^{2.9}+e^{2.1}} \end{bmatrix} = \begin{bmatrix} 0.68997 \\ 0.31003 \end{bmatrix}$$

$$L = -\log(0.68997) = 0.37110$$

$\frac{\partial E}{\partial E(1)}$:

$$\frac{\partial L}{\partial h^{(l)}} = \left[\frac{-1}{0.68997}, 0\right]$$

$$h^{(l)} = \begin{bmatrix} 0.68997 \\ 0.31003 \end{bmatrix} \quad (A)$$

$$\frac{\partial L}{\partial h^{(l-1)}} = \frac{\partial L}{\partial h^{(l)}} \frac{\partial h^{(l)}}{\partial h^{(l-1)}}$$

From (A):

$$= \left[\frac{-1}{0.68997}, 0\right] \begin{bmatrix} 0.68997(1-0.68997) & -0.68997(0.31003) \\ -0.31003(0.68997) & 0.31003(1-0.31003) \end{bmatrix}$$

$$= \left[-(1-0.68997), 0.31003\right]$$

$$\frac{\partial L}{\partial h^{(l-1)}} = \left[-0.31003, 0.31003\right]$$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial h_i^{(l-1)}} \left[h^{(l-2)}\right]^T \quad (B)$$

$$h^{(l-2)} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.2 \end{bmatrix}$$

From (B):

$$\frac{\partial L}{\partial w} = \begin{bmatrix} (-0.31003)[0.3 & 0.2 & 0.2] \\ (0.31003)[0.3 & 0.2 & 0.2] \end{bmatrix}$$

$$= \begin{bmatrix} -0.09301 & -0.06201 & -0.06201 \\ 0.09301 & 0.06201 & 0.06201 \end{bmatrix}$$

(iii)

$$\frac{\partial L}{\partial h^{(l-2)}} = \frac{\partial L}{\partial h^{(l-1)}} \cdot W$$

$$= [-0.31003,\ 0.31003] \begin{bmatrix} 5 & 4 & 3 \\ 3 & 4 & 2 \end{bmatrix}$$

$$= [-0.62005 \quad 0 \quad -0.31003]$$

(iv)

$$W' = W - \alpha \frac{\partial L}{\partial W}$$

$$= \begin{bmatrix} 5 & 4 & 3 \\ 3 & 4 & 2 \end{bmatrix} - 2 \begin{bmatrix} -0.09301 & -0.06201 & -0.06201 \\ 0.09301 & 0.06201 & 0.06201 \end{bmatrix}$$

$$= \begin{bmatrix} 5.181 & 4.124 & 3.124 \\ 2.814 & 3.876 & 1.876 \end{bmatrix}$$

[15 points]
Question 4: OpenCV and numpy

Given a 3 channel color square image:

```
img = cv2.imread('sample.jpg', cv2.IMREAD_COLOR)
img = cv2.resize(img, (100, 100))
```

Choose the best matches between OpenCV code and numpy code that perform the same functions. Briefly discuss the meaning of the corresponding codes.

OpenCV:

```
a/ cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
b/ cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
c/ cv2.warpAffine(img,
cv2.getRotationMatrix2D((W/2,H/2),90,1), (W,H)) # W, H
are width and height of the image
d/ cv2.getGaussianKernel(5, 1)
e/ cv2.threshold(img[:, :, 0], 40, 255,
cv2.THRESH_BINARY)
```

Numpy:

```
1/ img[:, :, np.array([0, 1, 2])] = img[:, :,
np.array([2, 1, 0])]

2/ img_new = np.zeros((img.shape[0], img.shape[1]),
dtype=np.uint8)
   img_new[img[:, :, 0] > 40] = 255

3/ np.around(img[:,:,0]*0.114 + img[:,:,1]*0.587 +
img[:,:,2]*0.299).astype(np.uint8)

4/ np.transpose(img, [1, 0, 2])

5/ x = np.array([-2, -1, 0, 1, 2])
   scale = np.sum(np.exp(-np.power(x,2)/2))
   np.exp(-np.power(x,2)/2)/scale
```

Answer:

OpenCV	Numpy	Meaning
a	3	Convert from color to greyscale
b	1	Convert BGR order to RGB
c	4	Rotate by 90 degrees
d	5	Obtain a 5x1 Gaussian kernel
e	2	If the first channel of img is larger than 40, it will become 255