Homework 3

Theory Part

Q1. Consider a CNN that has

1. Input of 14x14 with 30 channels.
2. A convolution layer C with 12 filters, each of size 4x4. The convolution zero padding is 1 and stride is 2, followed by a ReLU activation.
3. A max pooling layer P that is applied over each of the C's output feature map, using 3x3 receptive field and stride 2.

What is the total size of C's output feature map?

Convolution output size is given as

$$O_{size} = ceil((M + 2r - ksize + 1)/k)$$

where M is the size of the concerned dimension, r is padding and k is stride, and $ksize$ is kernel size of the corresponding dimension. Therefore,

$$O_{size} = ceil((14 + 2 * 1 - 4 + 1)/2)$$

In [1]:
```
import math

Osize = math.ceil((14 + 2*1 - 4 + 1)/2)
print(Osize)
```

```
7
```

So the total size of C's output feature map is 7x7.

What is the total size of P's output feature map?

Same formula, just without the padding.

In [2]:
```
Osize = math.ceil((7 - 3 + 1)/2)
print(Osize)
```

```
3
```

So the total size of O's output feature map is 3x3.

Now we want to compute the overhead of the above CNN in terms of floating point operation (FLOP). FLOP can be used to measure computer’s performance. A decent processor nowadays can perform in Giga-FLOPS, that means billions of FLOP per second. Assume the inputs are all scalars (we have 14 × 14 × 30 scalars as input), we have the computational cost of:

1. 1 FLOP for a single scalar multiplication xi · xj
2. 1 FLOP for a single scalar addition xi + xj
3. (n − 1) FLOPs for a max operation over n items: max{x1, ..., xn}

How many FLOPs layer C and P cost in total to do one forward pass?

For each cell of each filter in C's output feature map, there must necessarily be $ksize^2$ number of *multiplication* done for each of the input *channel* (for this 2d kernel).

$$C_{mul} = O_{size}^2 * channels * ksize^2 * filters$$
$$C_{mul} = 7^2 * 30 * 4^2 * 12$$
$$C_{mul} = 282240$$

In [3]:
```
7**2 * 30 * 4**2 * 12
```
Out[3]: 282240

For each cell of each filter in C's output feature map, there must also necessarily be $(ksize^2 - 1)$ number of *addition* (for this 2d kernel) to sum up all multiplication that is done for that output cell, for each of the input *channel* .

$$C_{add} = O_{size}^2 * (ksize^2 - 1) * filters * channel$$
$$C_{add} = 7^2 * (4^2 * 30 - 1) * 12$$
$$C_{add} = 281652$$

In [4]:
```
7**2 * (4**2 * 30 - 1) * 12
```
Out[4]: 281652

For each cell of P's output feature map, there must be a *max operation* operating on $ksize^2$ cells of the previous map.

$$P_{max} = O_{size}^2 * C_{channel} * (ksize^2 - 1)$$

$$P_{max} = 3^2 * 12 * (3^2 - 1)$$

$$P_{max} = 864$$

In [5]:
```
3**2 * 12 * (3**2 -1)
```
Out[5]: 864

so the total FLOP is,

$$T_{FLOP} = C_{mul} + C_{add} + P_{max}$$

$$T_{FLOP} = 282240 + 281652 + 864$$

$$T_{FLOP} = 564756$$

In [6]:
```
282240 + 281652 + 864
```
Out[6]: 564756

..

Q2 Refer to the neural network at *figure 1* with input $x \in R^1$. The activation function for z_1, z_2, and z_3 is the sigmoid function: $\frac{1}{1+e^{-w \cdot x}}$,

$$h(x) = \frac{1}{1+e^{-x}} \qquad (1)$$

$$z_1 = h(x \cdot w_{(x,1)}) \qquad (2)$$

$$z_2 = h(z_1 \cdot w_{(1,2)}) \qquad (3)$$

$$z_3 = h(z_1 \cdot w_{(1,3)}) \qquad (4)$$

For the error E, instead of using the softmax function we learned in class, we use the quadratic error function for regression purpose,

$$E = \sum_{i \,\epsilon\, data} \left((z_2 - y_{2i})^2 + (z_3 - y_{3i})^2 \right)$$

[**6p**] Write down an expression for the gradients of all three weights: $\frac{\partial E}{\partial w(x,1)}, \frac{\partial E}{\partial w(1,2)}, \frac{\partial E}{\partial w(1,3)}$.

Going backwards through the network,

$$\frac{\partial E}{\partial w(1,3)} = \sum_{i\,\epsilon\,data} \frac{\partial (z_2 - y_{2i})^2}{\partial w(1,3)} + \frac{\partial (z_3 - y_{3i})^2}{\partial w(1,3)}$$

$$\frac{\partial E}{\partial w(1,3)} = \sum_{i\,\epsilon\,data} \frac{\partial (z_3 - y_{3i})^2}{\partial w(1,3)}$$

$$\frac{\partial E}{\partial w(1,3)} = \sum_{i\,\epsilon\,data} 2(z_3 - y_{3i}) \cdot \frac{\partial z_3}{\partial w(1,3)}$$

$$\frac{\partial E}{\partial w(1,3)} = \sum_{i\,\epsilon\,data} 2(z_3 - y_{3i}) \cdot \frac{\partial h(z_1 \cdot w_{(1,3)})}{\partial w(1,3)}$$

$$since \qquad \frac{\partial h(x)}{\partial (x)} = h(x) \cdot (1 - h(x)), \qquad < sigmoid >$$

$$\frac{\partial E}{\partial w(1,3)} = \sum_{i\,\epsilon\,data} 2(z_3 - y_{3i}) \cdot h(z_1 \cdot w_{(1,3)}) \cdot (1 - h(z_1 \cdot w_{(1,3)})) \cdot \frac{\partial (z_1 \cdot w_{(1,3)})}{\partial w(1,3)}$$

$$\frac{\partial E}{\partial w(1,3)} = \sum_{i\,\epsilon\,data} 2(z_3 - y_{3i}) \cdot h(z_1 \cdot w_{(1,3)}) \cdot (1 - h(z_1 \cdot w_{(1,3)})) \cdot z_1$$

Likewise,

$$\frac{\partial E}{\partial w(1,2)} = \sum_{i\,\epsilon\,data} 2(z_2 - y_{2i}) \cdot h(z_1 \cdot w_{(1,2)}) \cdot (1 - h(z_1 \cdot w_{(1,2)})) \cdot z_1$$

As for $w_{(x,1)}$,

$$\frac{\partial E}{\partial w(x,1)} = \sum_{i\,\epsilon\,data} \frac{\partial (z_2 - y_{2i})^2}{\partial w(x,1)} + \frac{\partial (z_3 - y_{3i})^2}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} \frac{\partial (z_a - y_{ai})^2}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai}) \frac{\partial (z_a - y_{ai})}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai}) \frac{\partial (z_a)}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai}) \frac{\partial (h(z_1 \cdot w_{(1,a)}))}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai})(h(z_1 \cdot w_{(1,a)})(1 - h(z_1 \cdot w_{(1,a)}) \frac{\partial (z_1 \cdot w_{(1,a)})}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai})(z_a)(1 - z_a) \frac{\partial (z_1 \cdot w_{(1,a)})}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai})(z_a)(1 - z_a)(w_{(1,a)}) \frac{\partial (z_1)}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai})(z_a)(1-z_a)(w_{(1,a)}) \frac{\partial h(x_i \cdot w_{(x,1)})}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai})(z_a)(1-z_a)(w_{(1,a)})(h(x_i \cdot w_{(x,1)}))(1-h(x_i \cdot w_{(x,1)})) \frac{\partial (x_i \cdot w_{(x,1)})}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai})(z_a)(1-z_a)(w_{(1,a)})(z_1)(1-z_1) \frac{\partial (x_i \cdot w_{(x,1)})}{\partial w(x,1)}$$

$$\frac{\partial E}{\partial w(x,1)} = \sum_{a\,\epsilon\,(2,3)} \sum_{i\,\epsilon\,data} 2(z_a - y_{ai})(z_a)(1-z_a)(w_{(1,a)})(z_1)(1-z_1) \cdot x_i$$

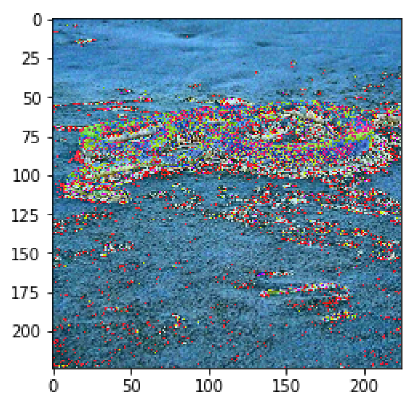
Coding Part

In [7]:
```
import torch
from wk3_homework import *
import matplotlib.pyplot as plt
from torchvision import transforms
```

In [8]:
```
dataset_count = 100
root_path = '../datasets/imagenet_first2500/'
data_singlecrop = Wk3Dataset(root_path, data_limit=dataset_count, five_crop=False)

# testing Dataset subclass - single center crop
first_img = data_singlecrop[0]
print(first_img['label'], data_singlecrop.classes[first_img['label']])
image = transforms.ToPILImage()(first_img['image'])
plt.imshow(image)
```

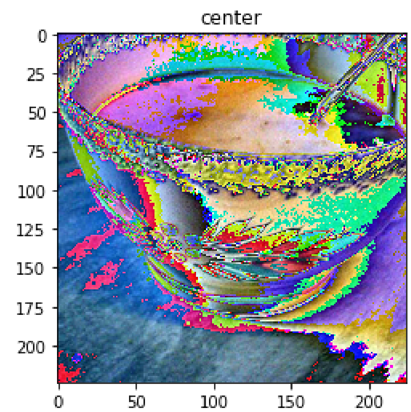
```
65 sea snake
```

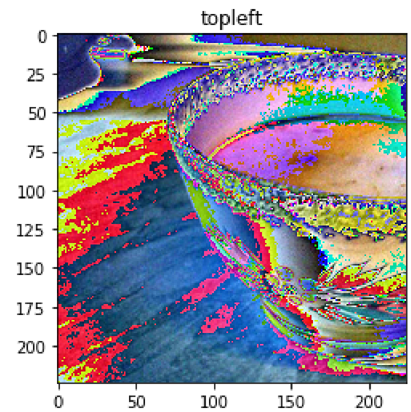
Out[8]: `<matplotlib.image.AxesImage at 0x1f6838951d0>`

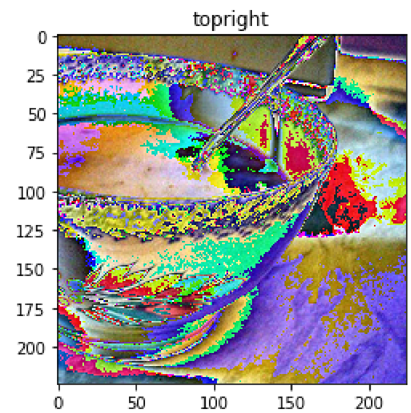


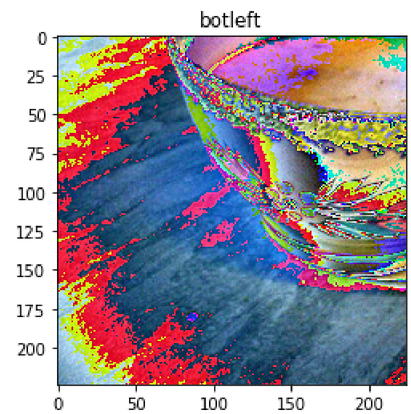
In [9]:
```
dataset_count = 100
root_path = '../datasets/imagenet_first2500/'
data_fivecrop = Wk3Dataset(root_path, data_limit=dataset_count, five_crop=True)

# testing Dataset subclass - single center crop
first_img = data_fivecrop[3]
print(first_img['label'], data_fivecrop.classes[first_img['label']])

pos = ['center', 'topleft', 'topright','botleft', 'botright']
for i in range(5):
    image = transforms.ToPILImage()(first_img['image'][i])
    plt.figure()
    plt.title(pos[i])
    plt.imshow(image)
```

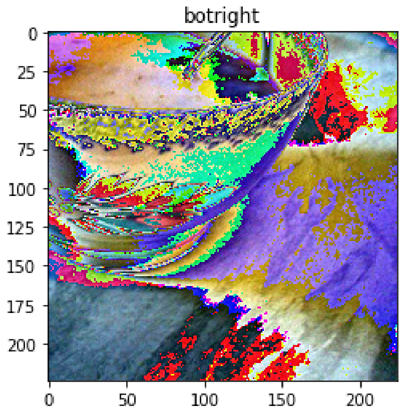
```
809 soup bowl
```











In [10]:
```
# running validation with normal centercrop
valset, model = run_validation(five_crop=False, dataset_count=250)
```

```
dataset length 250 torch.Size([3, 224, 224])
Val - Epoch 0..
      >> Epoch loss 0.33772 accuracy 0.664                      in 8.3146s
```

In [11]:
```
# running validation with fivecrop
valset, model = run_validation(five_crop=True, dataset_count=250)
```

```
dataset length 250 torch.Size([5, 3, 224, 224])
Val - Epoch 0..
      >> Epoch loss 0.31602 accuracy 0.692                      in 21.3460s
```