第 4 章课后习题参考答案

1. >>> data=np.ones(shape=5)
>>> data[4]=3
>>> print(data)
array([1., 1., 1., 1., 3.])

2. >>> data = np.zeros((3, 3))
>>> print(data)
array([[0., 0., 0.],
[0., 0., 0.],
[0., 0., 0.]])
>>> print('%d bytes'%(data.size * data.itemsize))
72 bytes

3. >>> np.arange(6, 37)
array([6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36])

4. >>> data = np.empty((3, 3))
>>> print(data)
array([[0.00000000e+000, 0.00000000e+000, 0.00000000e+000],
[0.00000000e+000, 0.00000000e+000, 4.78255545e-321],
[1.62122812e-086, -1.82703838e-311, 6.52127111e-095]])
>>> print(data.dtype.name)
float64

5. >>> data = np.random.random((3, 3))
>>>datamin, datamax= data.min(), data.max()
>>> print(data)
array([[0.98106899, 0.68464983, 0.43050183],
[0.74510984, 0.57620886, 0.48391251],
[0.13327387, 0.33302911, 0.82532384]])
>>> print(datamin, datamax)
0.13327387178172 0.9810689942444079

6. >>> data=np.arange(5)

```
>>> print(data)
[0 1 2 3 4]
>>> data=data[::-1]
>>> print(data)
[4 3 2 1 0]
```

7.
```
>>> data=np.arange(16).reshape(4, 4)
>>> print(data)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9  10  11]
 [12  13  14  15]]
>>> print(data[2, 0])      //获取第三行第一列的数据
8
>>> print(data[2, 1])//获取第三行第二列的数据
9
```

8.
```
>>> data=np.arange(16).reshape(4, 4)
>>> print(data[:2, 1:3])
    [[1  2]
     [5  6]]
```

9.
```
>>> arr1=np.arange(8).reshape(2, 2, 2)
>>> print(arr1)
[[[0  1]
  [2  3]]

 [[4  5]
  [6  7]]]
>>> print(np.sqrt(arr1))
[[[0.          1.        ]
  [1.41421356  1.73205081]]

 [[2.          2.23606798]
  [2.44948974  2.64575131]]]
```

```
>>>arr2=np.arange(-2, 6).reshape(2, 2, 2)
>>> print(arr2)
```

```
[[[-2 -1]
  [ 0  1]]

 [[ 2  3]
  [ 4  5]]]
>>> print(np.square(arr2))
[[[ 4  1]
  [ 0  1]]

 [[ 4  9]
  [16 25]]]
>>> print(np.append(arr1, arr2))
[ 0  1  2  3  4  5  6  7  -2  -1  0  1  2  3  4  5]
10. >>> arr1 = np.random.random([3,  3])
>>> print(arr1)
[[0.37547457  0.38921484  0.1632189 ]
[0.4627367  0.5782657  0.11194791]
[0.29548517  0.37633842  0.02405697]]
>>>arr2=np.random.randint(50, 100, (3, 3))
>>> print(arr2)
[[82  89  62]
 [83  92  76]
 [96  75  93]]
>>> print('两个数组相加:', np.add(arr1, arr2))
两个数组相加:[[82.37547457  89.38921484  62.1632189 ]
[83.4627367  92.5782657  76.11194791]
[96.29548517  75.37633842  93.02405697]]
>>> print('两个数组相乘:', np.multiply(arr1, arr2))
两个数组相乘:[[30.78891451  34.64012094  10.11957173]
              [38.407146  53.200444  8.50804084]
              [28.36657654  28.22538135  2.23729857]]
```

11. 见 4.2.1 节

12. 见 4.2.2 节

13. 常见的池化方式包括最大池化、平均池化、全局最大池化。

14. 卷积神经网络的主要结构包括输入层、卷积层、池化层、激活层、全连接层以及输出层。

15. 在全局看来，感受野是卷积神经网络的每一层输出的特征图(feature map)上的像素点在输入图片上映射的区域大小。在局部看来，感受野是卷积神经网络每一层输出的特

征图(feature map)上的像素点在上一层映射的区域大小。通俗点说，就是特征图上的一个像素点对应输入图像的区域。

16. 略

17. 见 4.3.3 节、4.3.5 节

18. 见 4.3.4 节

19. 完全卷积网络(FCN)用于像素级分类，不需要提取 patch，从而避免了计算冗余。U-Net 建立在 FCN 的概念上，但通过沿着扩展路径使用一系列上采样层来生成与原始图像相同分辨率的分割地图，U-Net 沿着扩展路径重复地将不同层次的特征通道串联起来，并在扩展路径的最后一个阶段基于大量的特征通道生成最终的分割图。

20. 略