ECE-7650 Assignment #1

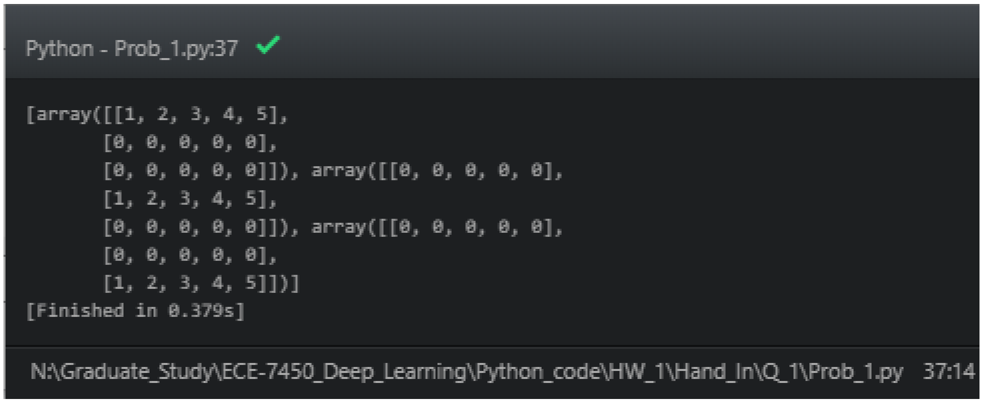
Student: Junyao Pu

Student #:7843296

Q1:

a) There will be L dimensions of dv/dW, and each dimension will have L*D elements.

b) The python code is attached (Prob_1.py), I was able to write this function by only one for loop. Below is the output of the python code.



Python - Prob_1.py:37 ✓

```
[array([[1, 2, 3, 4, 5],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0]]), array([[0, 0, 0, 0, 0],
       [1, 2, 3, 4, 5],
       [0, 0, 0, 0, 0]]), array([[0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [1, 2, 3, 4, 5]])]
[Finished in 0.379s]
```

N:\Graduate_Study\ECE-7450_Deep_Learning\Python_code\HW_1\Hand_In\Q_1\Prob_1.py   37:14

c) Unfinished

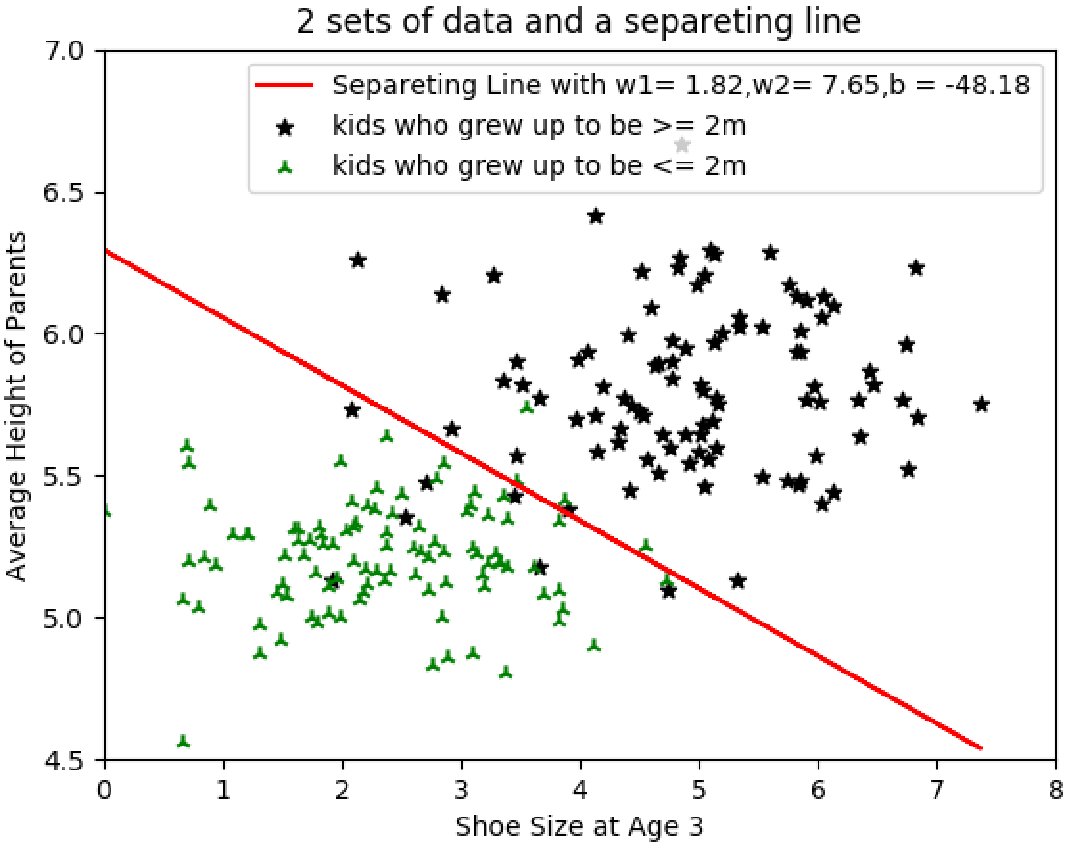
Q2:

a): Write a function that computer the cost function, the python code is attached (Prob_2_part_abc.py)

b): Write a function that computer the gradient of cost function, the python code is attached (Prob_2_part_abc.py)

c): Write a sigmoid function, the python code is attached (Prob_2_part_abc.py)

d): Run gradient descent algorithm, the python code is attached (Prob_2_part_abc.py). Below is the plot of the data and separating line with trained parameters. The separating line is trained with 0.1 training rate, 50000 training steps, those trained parameters are w1 = 1.82, w2 = 7.65 and b = -48.18.



Q3:

a): The python code is attached (Prob_3_part_a.py). I trained 10 two-class classifiers with data_batch_1 training set, I used training rate 0.001 with 200 training steps. I end with around 90% accuracy for all 10 classifiers. Those trained weights and bias are saved in csv file (weights.csv and bias.csv are attached) for calculation of part b and c.

b): The python code is attached (Prob_3_part_b.py). Below are those image versions of weights for each class.

Can you interpret these weight vector ….?

Those weight vector basically is a filter that doing the dot product with the image. What they are doing is looking for the object on the image, if there is the correct object on the image, the dot product of the filter and image will most likely give good prediction of the class.

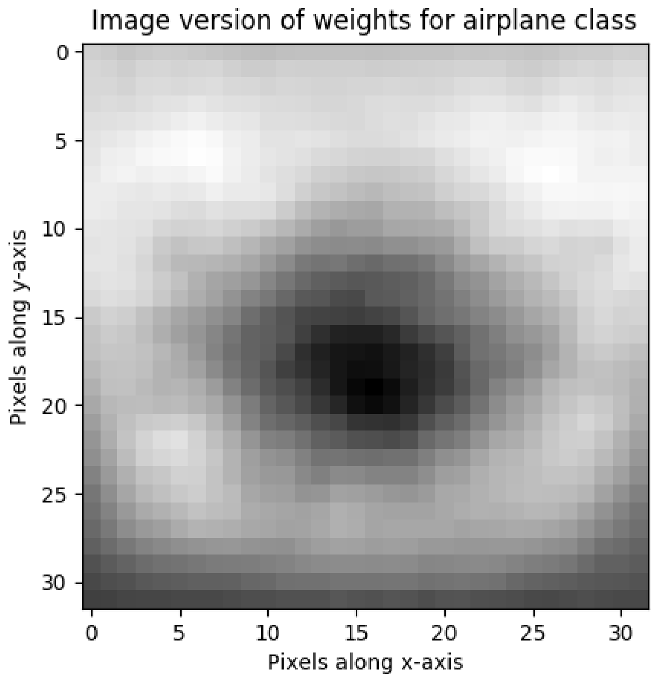


Image version of weights for airplane class

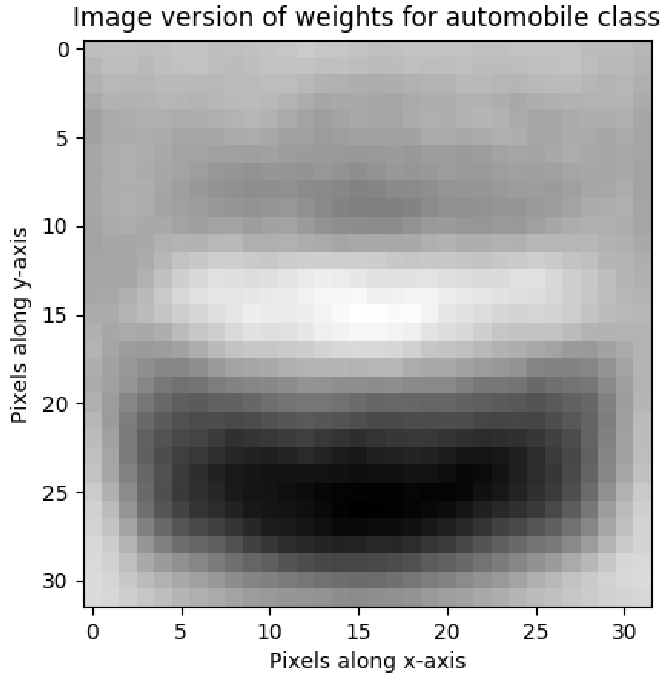


Image version of weights for automobile class

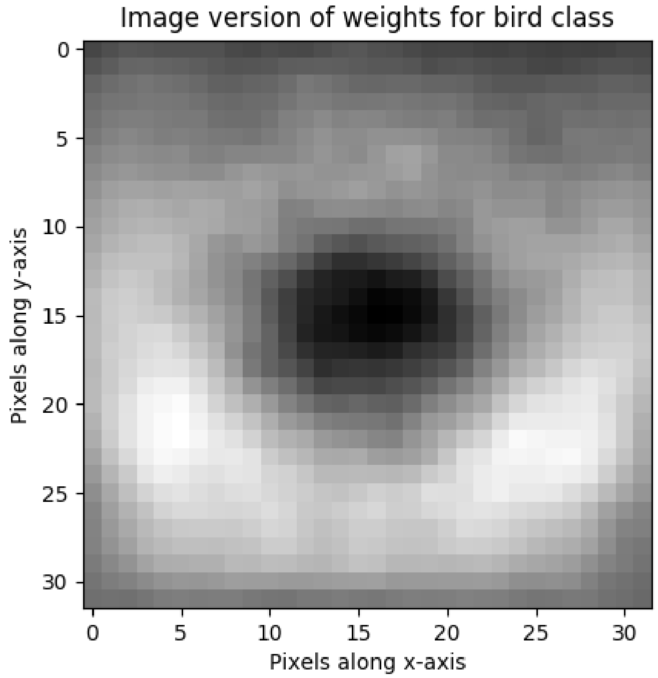


Image version of weights for bird class
Pixels along y-axis
0
5
10
15
20
25
30
0
5
10
15
20
25
30
Pixels along x-axis

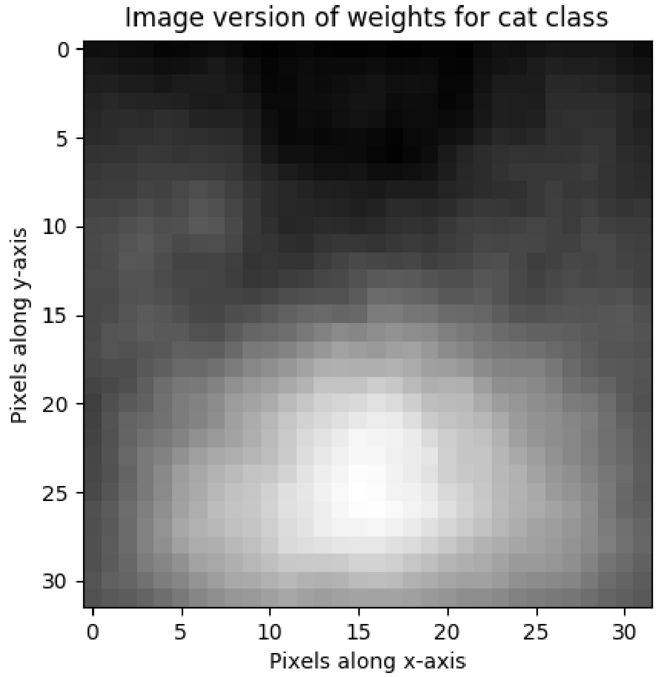


Image version of weights for cat class
Pixels along y-axis
0
5
10
15
20
25
30
0
5
10
15
20
25
30
Pixels along x-axis

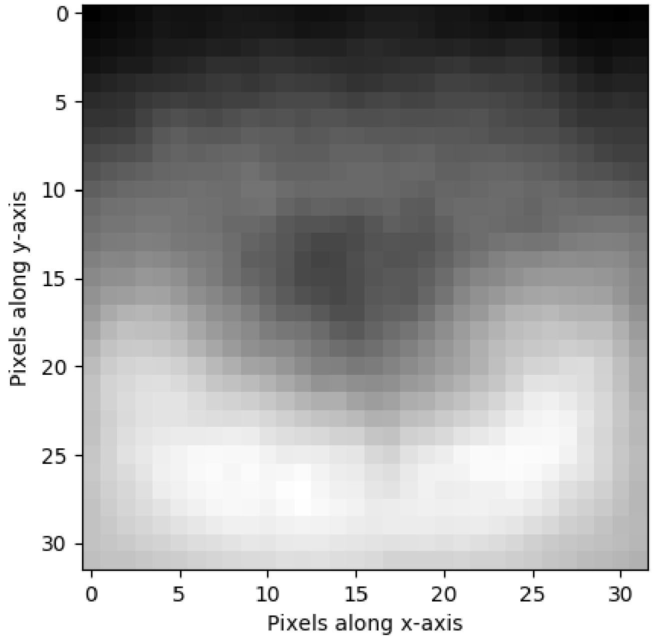


Image version of weights for deer class

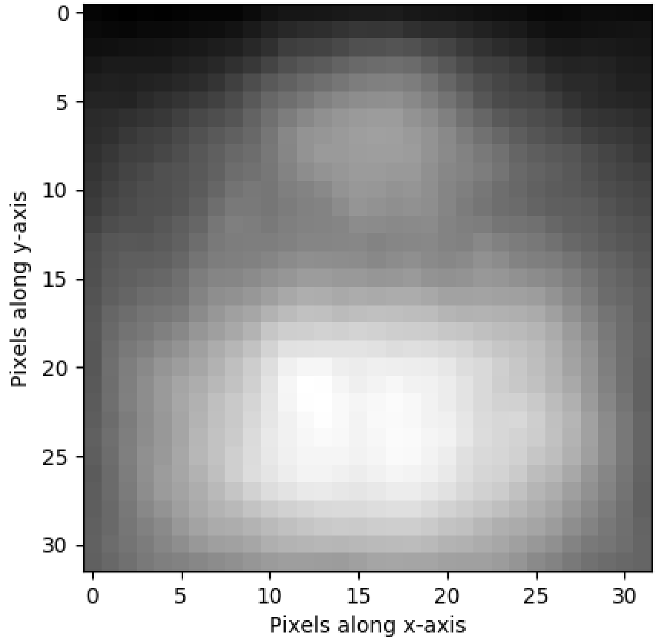


Image version of weights for dog class

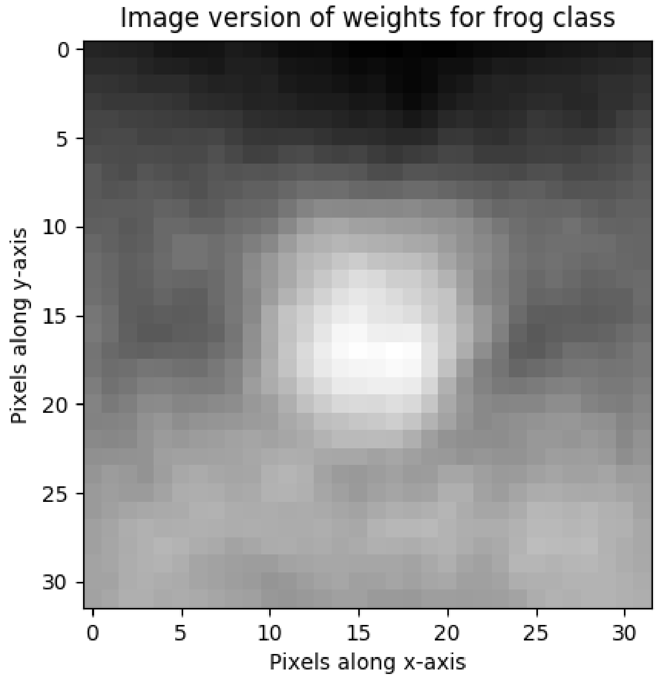


Image version of weights for frog class

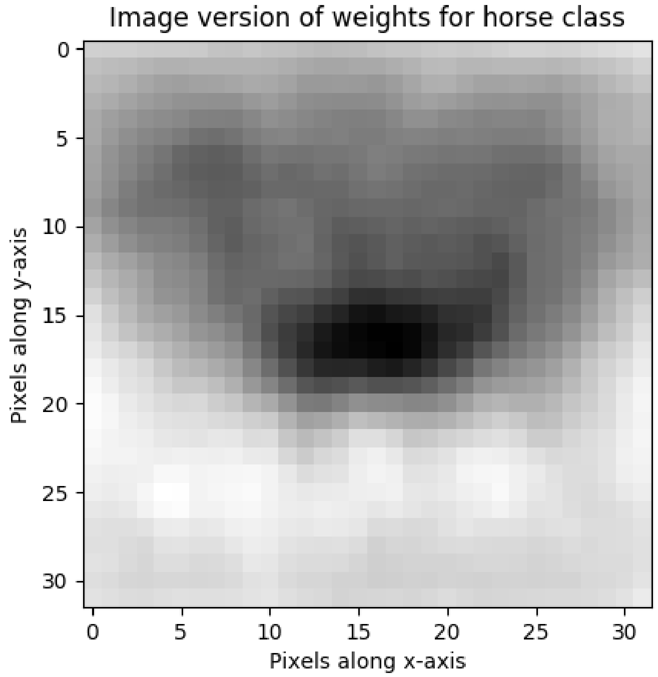


Image version of weights for horse class

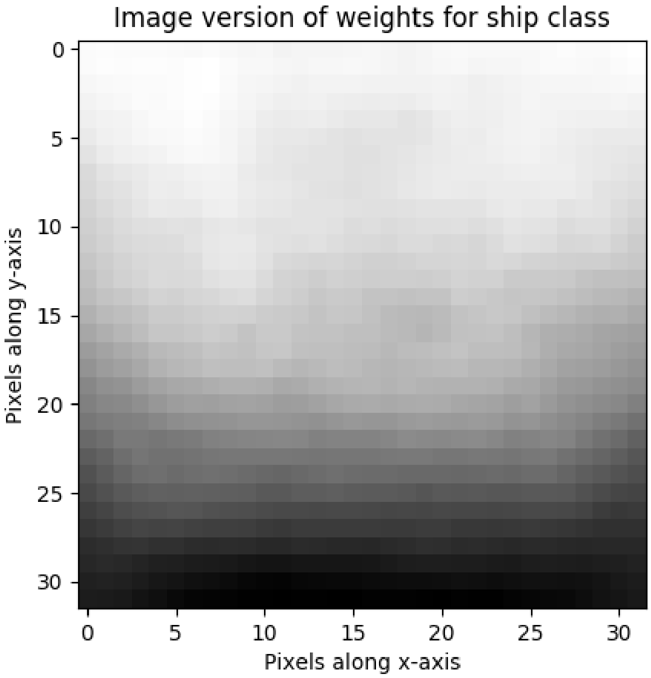


Image version of weights for ship class

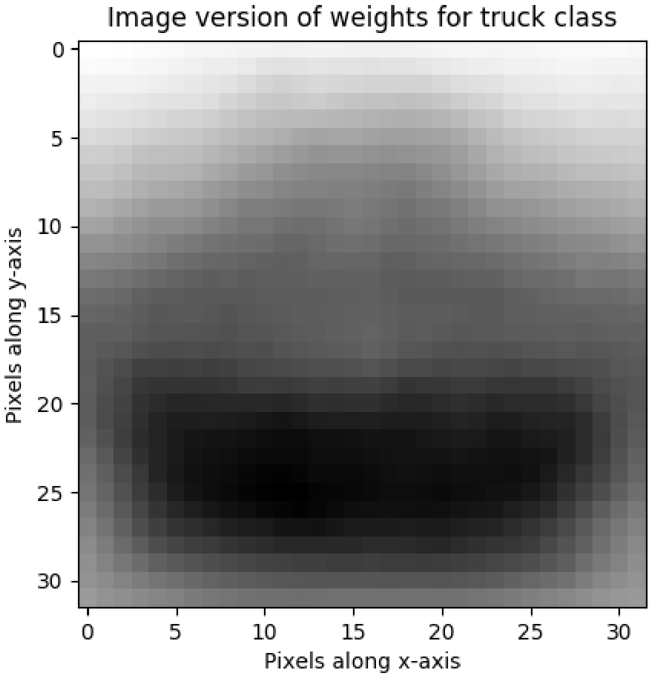
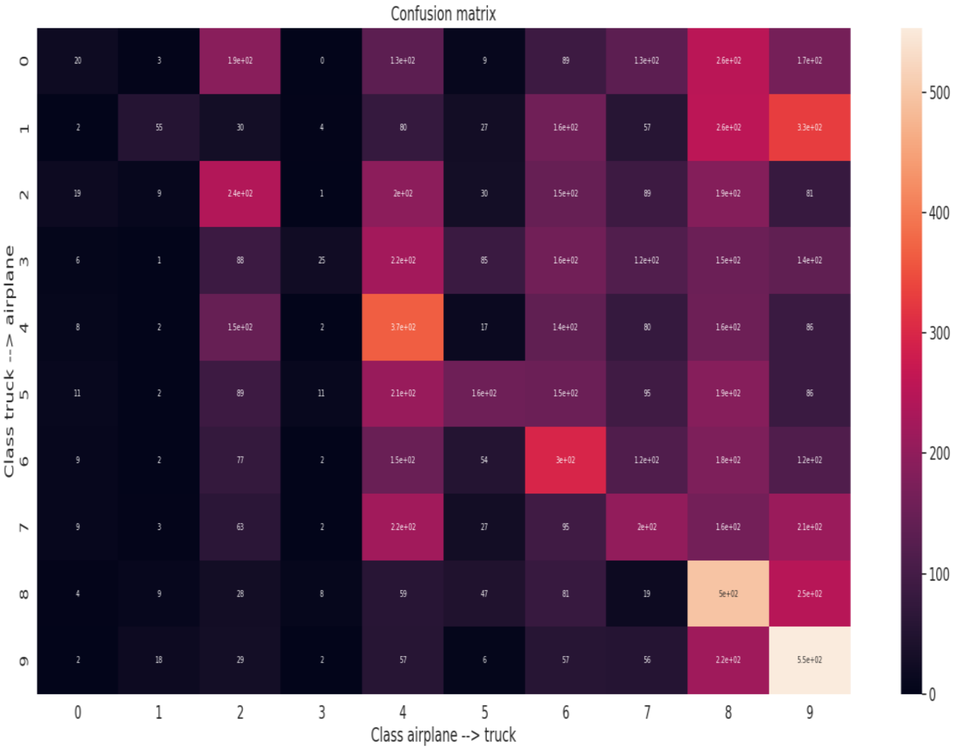


Image version of weights for truck class c): The python code is attached (Prob_3_part_c.py). Below is the confusion matrix where the class 0 to 9 on the x and y axis are 'airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck'. The performance of class 2, 4, 6, 7, 8 and 9 are batter than anothers base on the confusion matrix.



Confusion matrix

	0	1	2	3	4	5	6	7	8	9
0	20	3	1.9e+02	0	1.3e+02	9	89	1.3e+02	2.6e+02	1.7e+02
1	2	55	30	4	80	27	1.6e+02	57	2.6e+02	3.3e+02
2	19	9	2.4e+02	1	2e+02	30	1.5e+02	89	1.9e+02	81
3	6	1	88	25	2.2e+02	85	1.6e+02	1.2e+02	1.5e+02	1.4e+02
4	8	2	1.5e+02	2	3.7e+02	17	1.4e+02	80	1.6e+02	86
5	11	2	89	11	2.1e+02	1.6e+02	1.5e+02	95	1.9e+02	86
6	9	2	77	2	1.5e+02	54	3e+02	1.2e+02	1.8e+02	1.2e+02
7	9	3	63	2	2.2e+02	27	95	2e+02	1.6e+02	2.1e+02
8	4	9	28	8	59	47	81	19	5e+02	2.5e+02
9	2	18	29	2	57	6	57	56	2.2e+02	5.5e+02