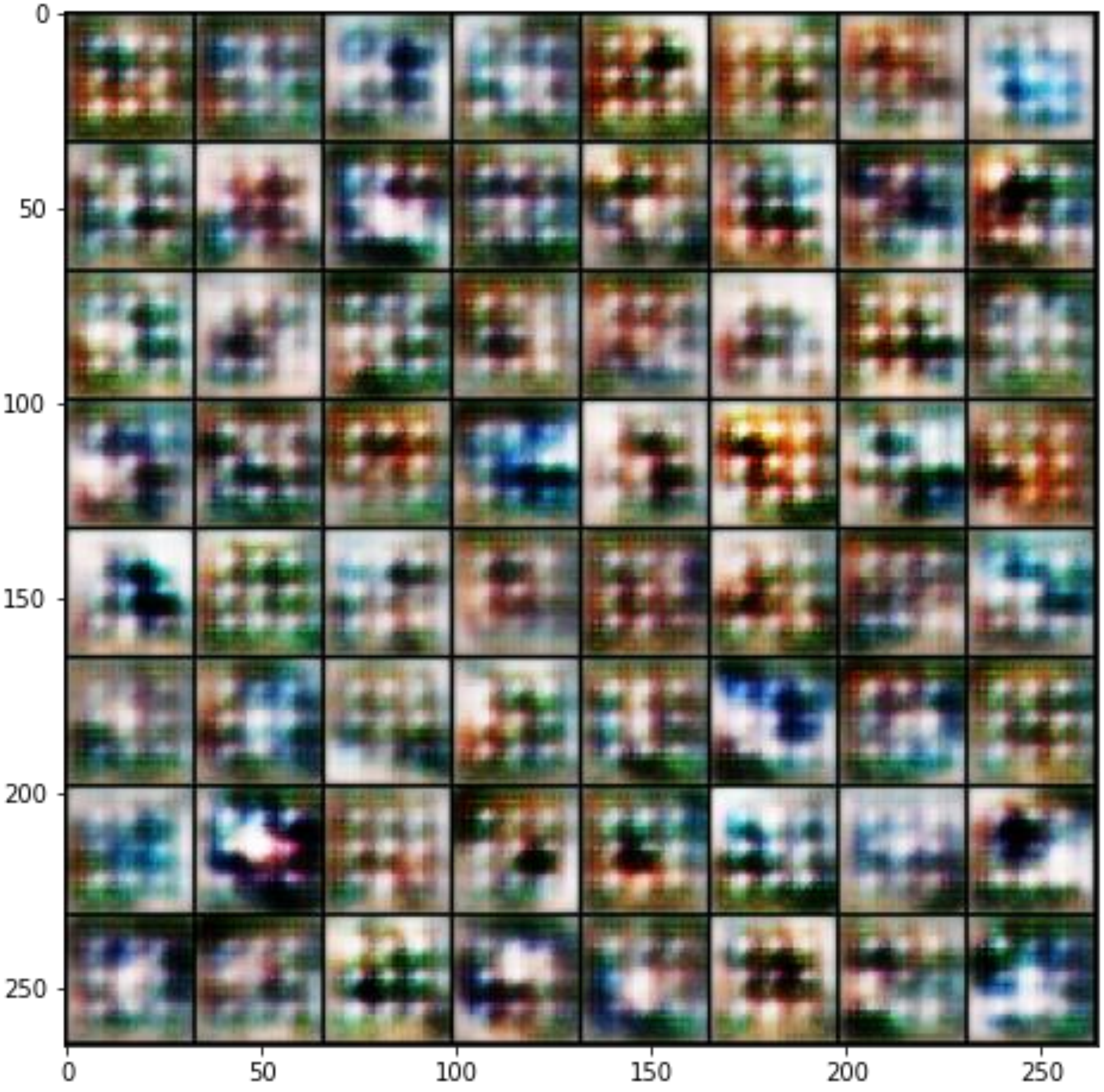
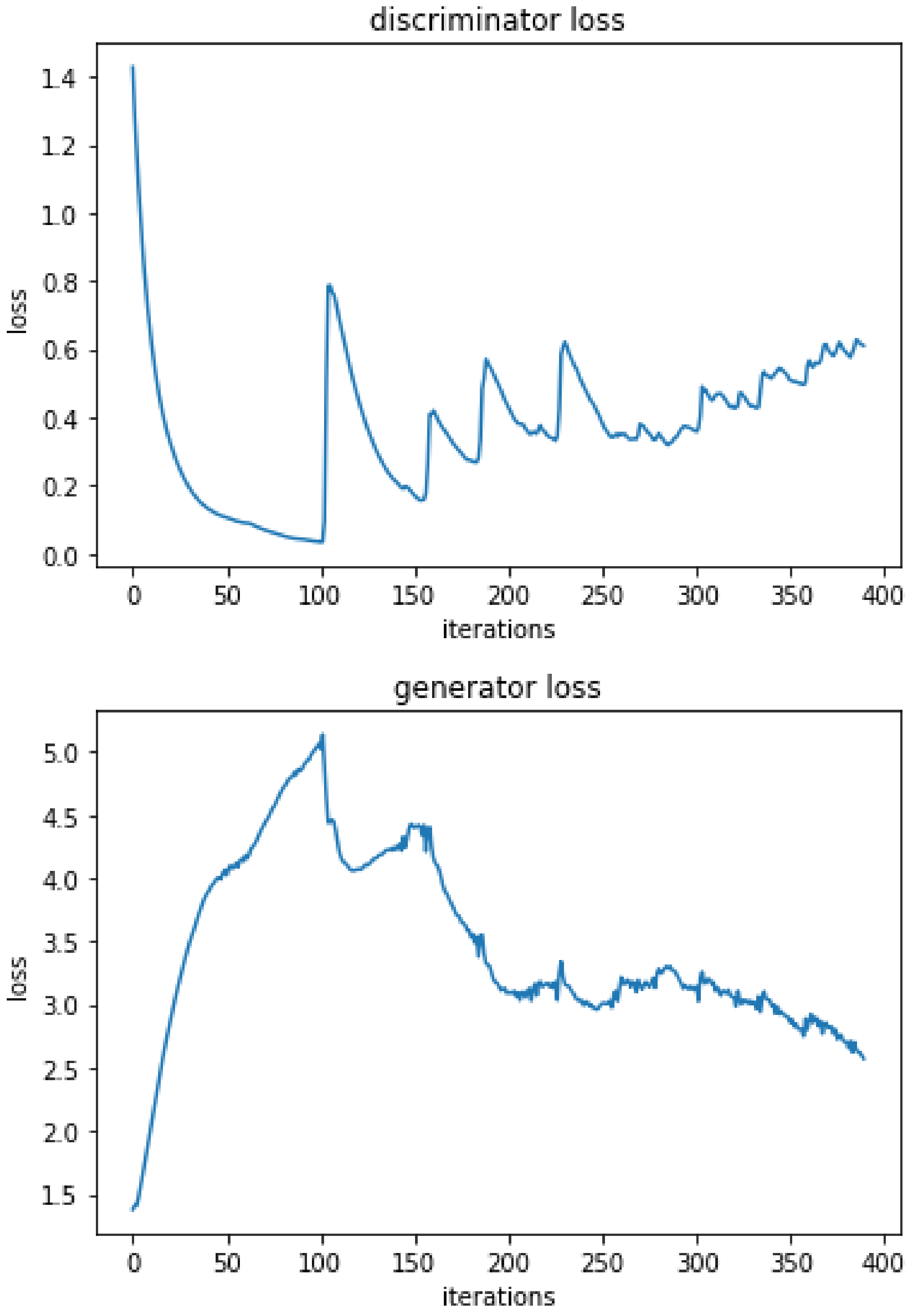
CSCI 566 Assignment 2

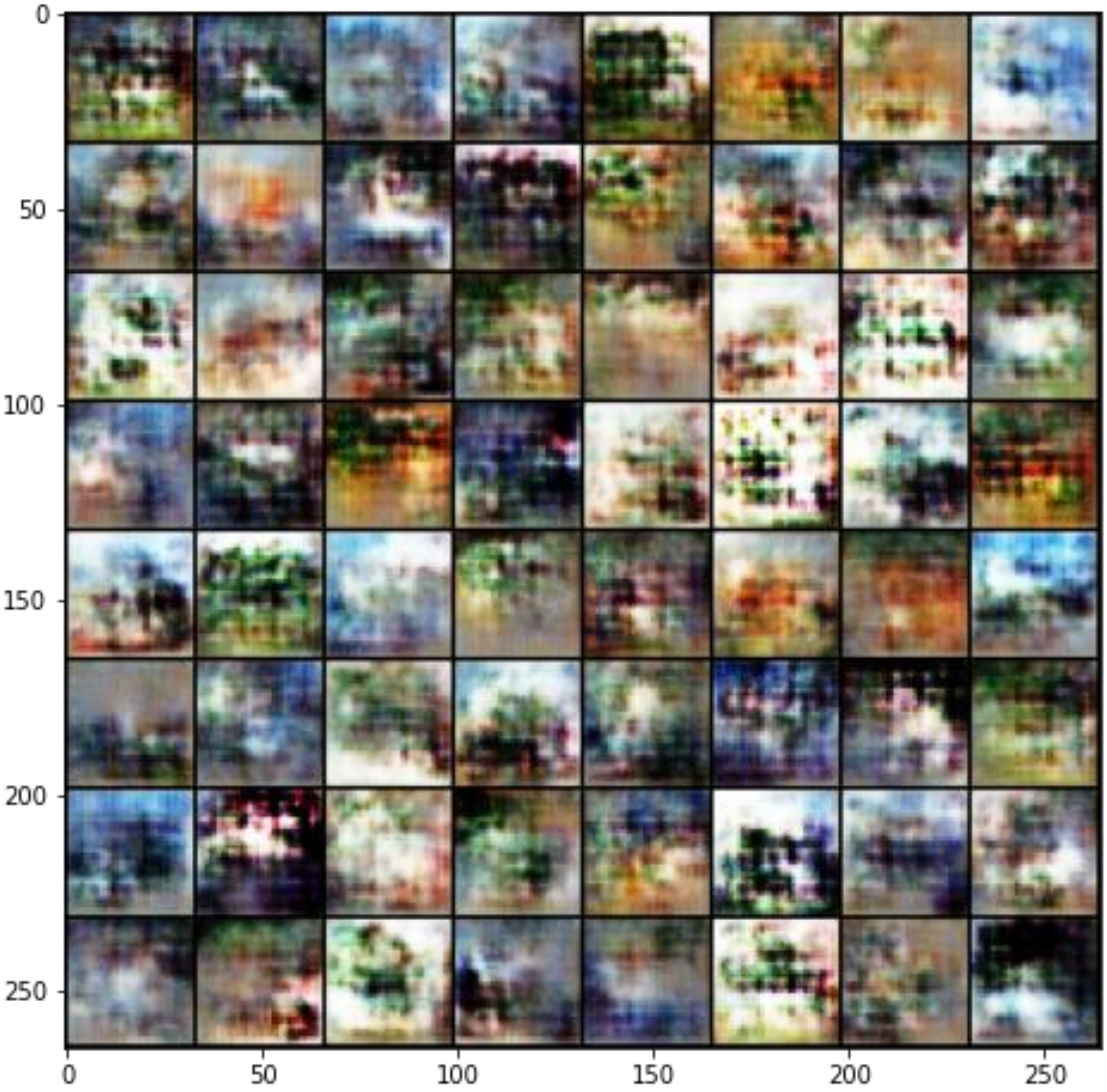
Problem 2

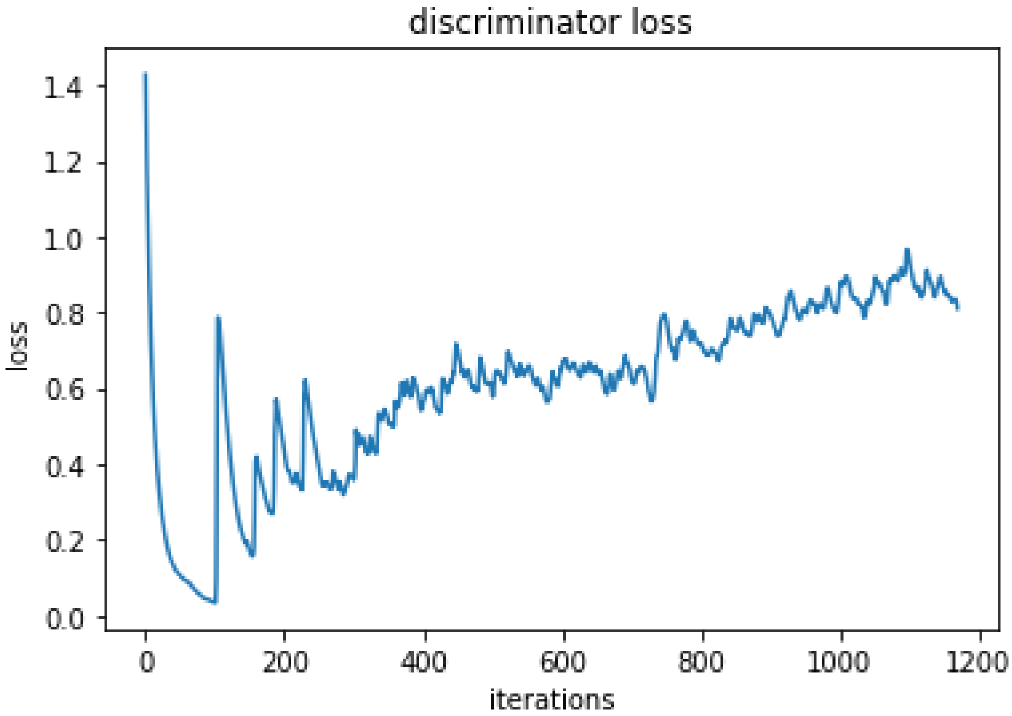
```
Start training ...
Iteration 100/9750: dis loss = 0.0183, gen loss = 5.5318
Iteration 200/9750: dis loss = 0.2393, gen loss = 2.3493
Iteration 300/9750: dis loss = 0.3297, gen loss = 2.9805
```

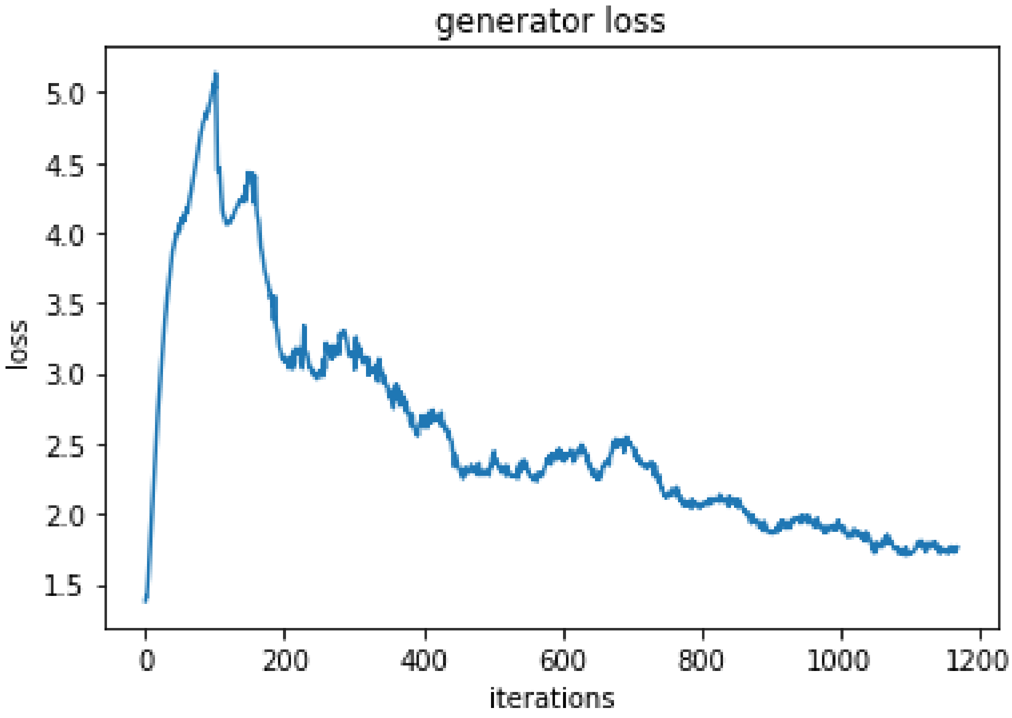




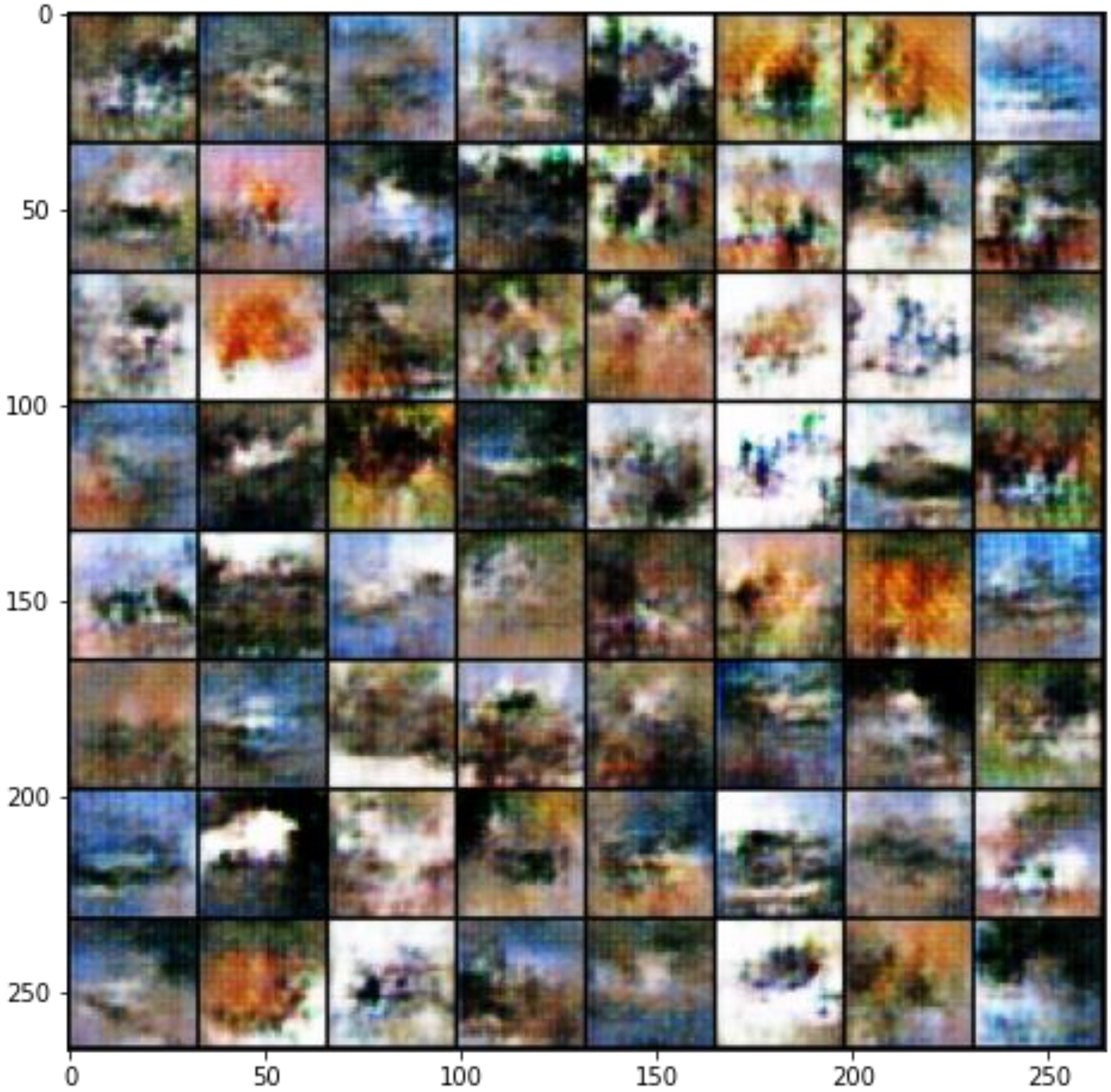
```
Iteration 400/9750: dis loss = 0.6466, gen loss = 1.8876
Iteration 500/9750: dis loss = 0.3383, gen loss = 3.2880
Iteration 600/9750: dis loss = 0.5232, gen loss = 2.6444
Iteration 700/9750: dis loss = 0.4147, gen loss = 2.3606
Iteration 800/9750: dis loss = 0.7587, gen loss = 1.5771
Iteration 900/9750: dis loss = 0.7117, gen loss = 1.6435
Iteration 1000/9750: dis loss = 1.3006, gen loss = 1.0956
Iteration 1100/9750: dis loss = 0.7966, gen loss = 1.5418
```

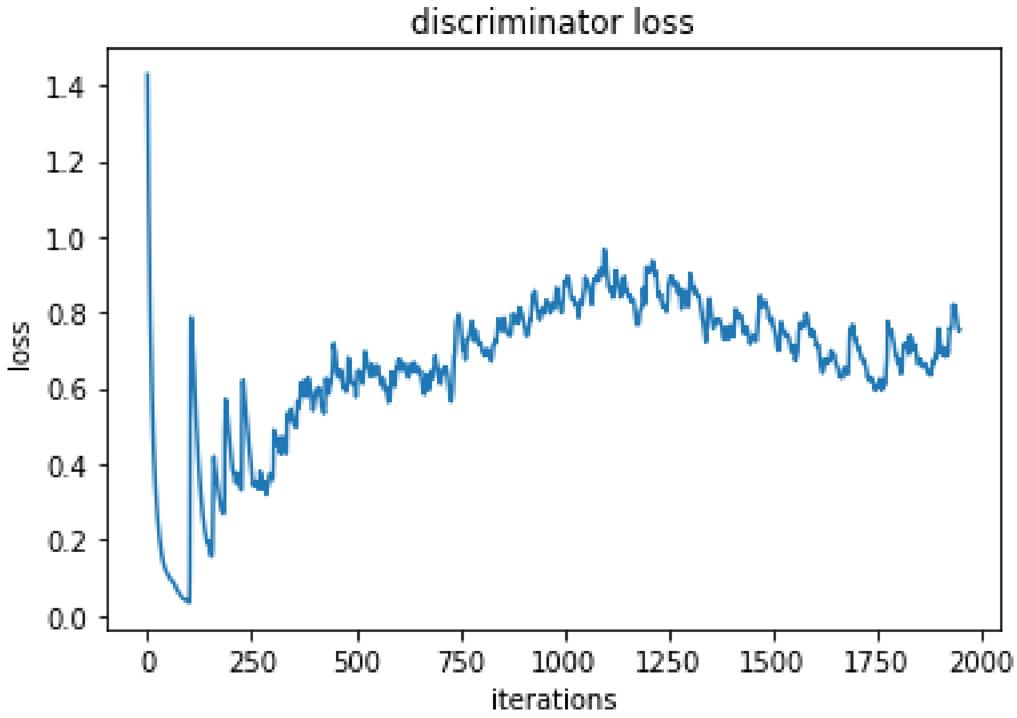

0
50
100
150
200
250
0
50
100
150
200
250

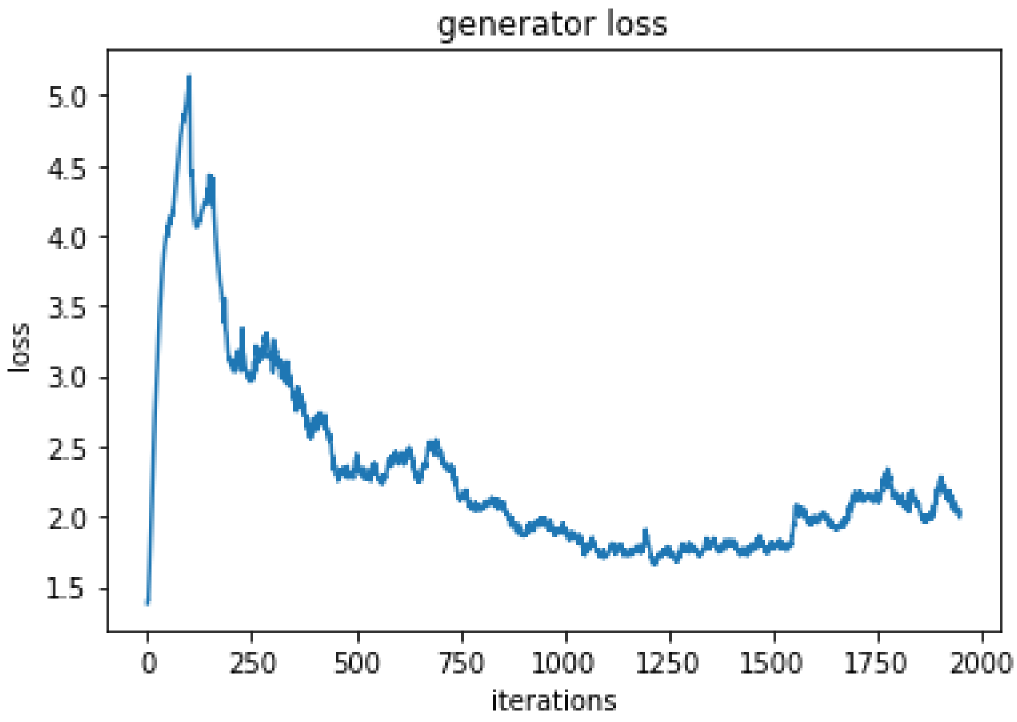




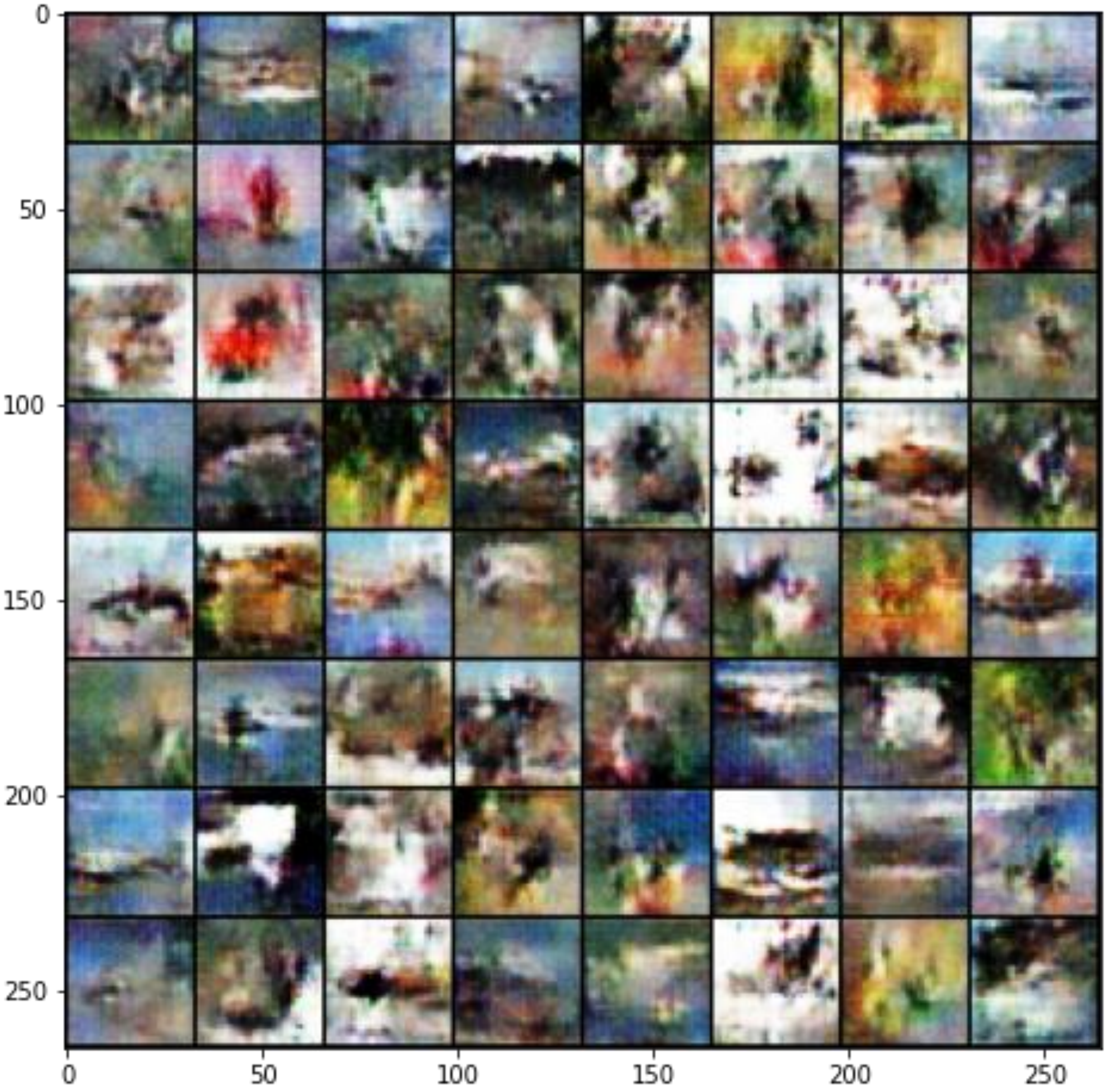
```
Iteration 1200/9750: dis loss = 0.8408, gen loss = 2.1036
Iteration 1300/9750: dis loss = 1.0077, gen loss = 1.0075
Iteration 1400/9750: dis loss = 1.0155, gen loss = 2.4543
Iteration 1500/9750: dis loss = 0.8537, gen loss = 2.2609
Iteration 1600/9750: dis loss = 0.7251, gen loss = 1.8889
Iteration 1700/9750: dis loss = 0.8608, gen loss = 1.2923
Iteration 1800/9750: dis loss = 0.5009, gen loss = 2.2207
Iteration 1900/9750: dis loss = 0.8170, gen loss = 2.9728
```

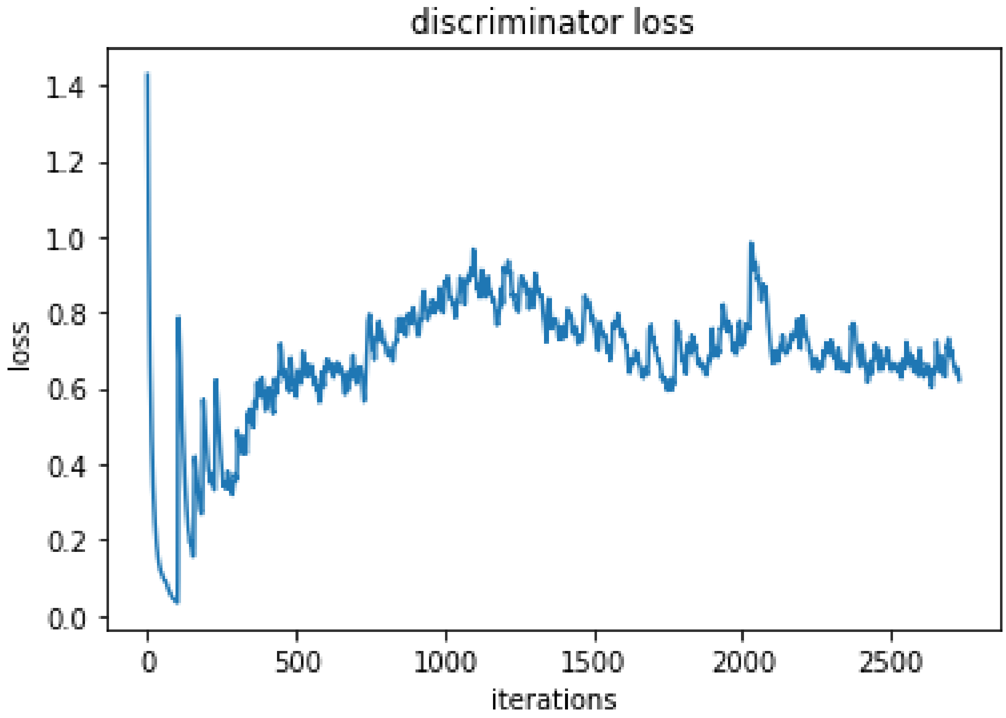

0
50
100
150
200
250
0
50
100
150
200
250

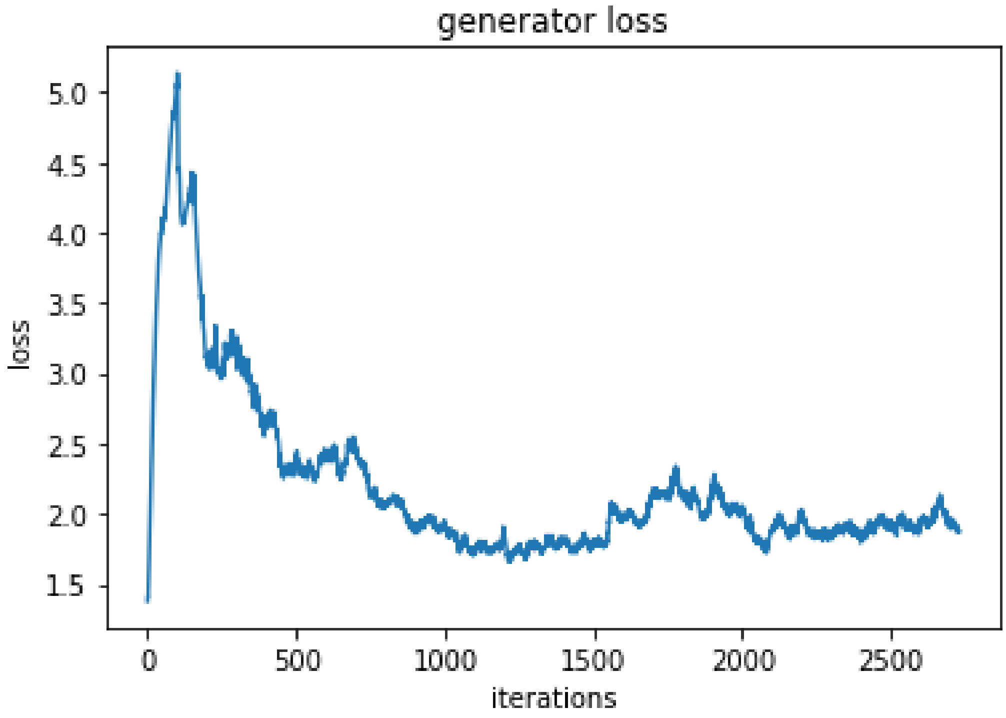




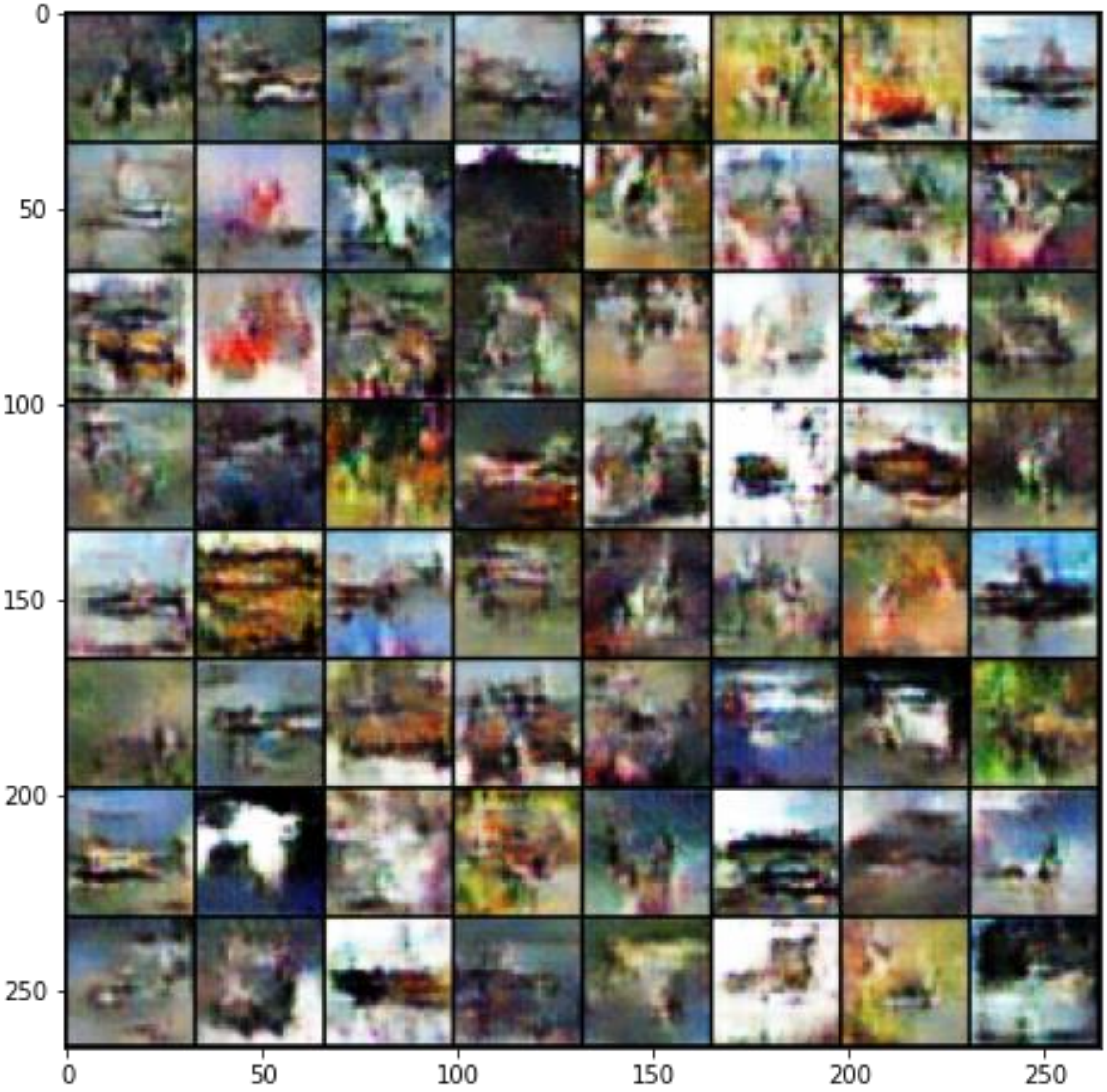
```
Iteration 2000/9750: dis loss = 0.6691, gen loss = 3.1559
Iteration 2100/9750: dis loss = 0.5777, gen loss = 1.9208
Iteration 2200/9750: dis loss = 1.1604, gen loss = 1.9801
Iteration 2300/9750: dis loss = 0.5320, gen loss = 1.6425
Iteration 2400/9750: dis loss = 1.3783, gen loss = 0.8878
Iteration 2500/9750: dis loss = 0.7680, gen loss = 1.3239
Iteration 2600/9750: dis loss = 0.5348, gen loss = 2.8684
Iteration 2700/9750: dis loss = 0.5930, gen loss = 2.4041
```

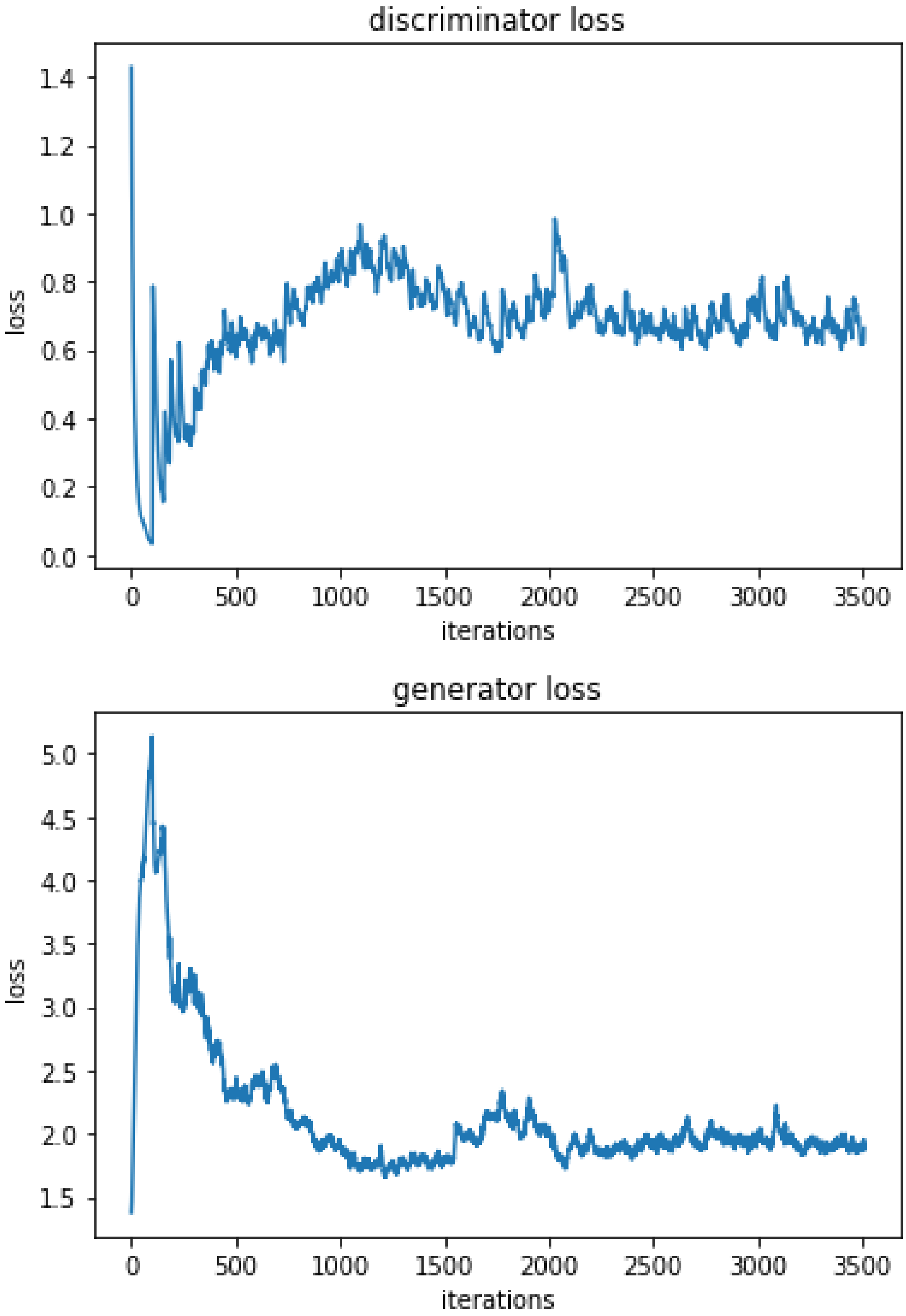

0
50
100
150
200
250
0
50
100
150
200
250



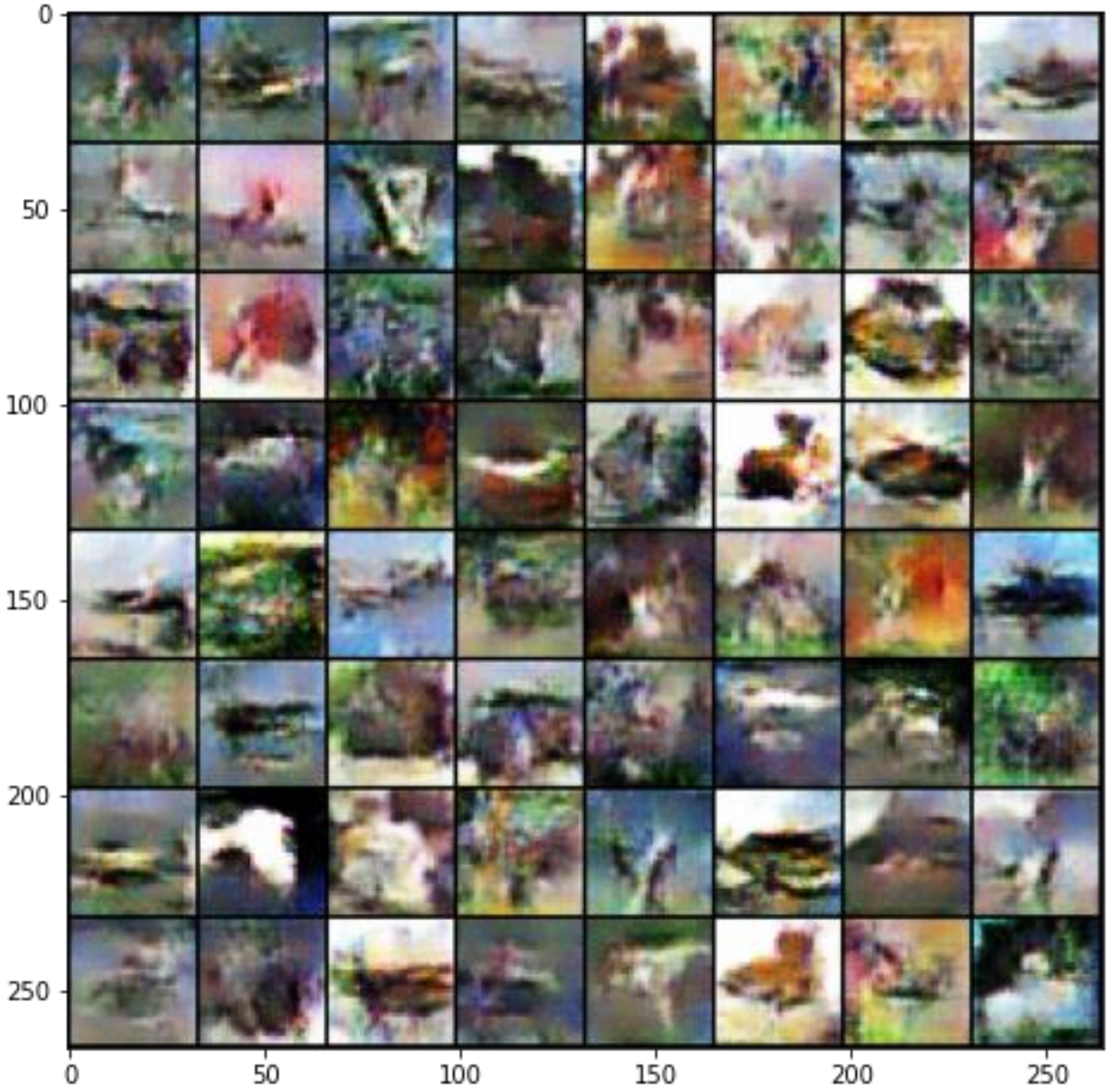


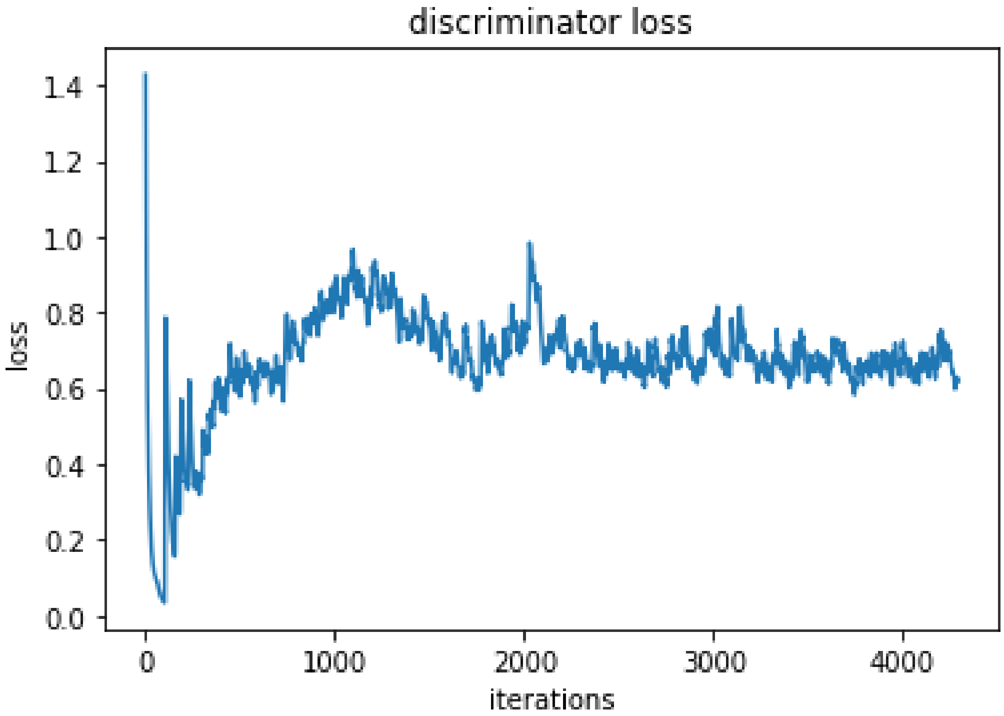
```
Iteration 2800/9750: dis loss = 0.5905, gen loss = 3.0138
Iteration 2900/9750: dis loss = 0.5535, gen loss = 2.5038
Iteration 3000/9750: dis loss = 0.4513, gen loss = 2.2995
Iteration 3100/9750: dis loss = 0.6145, gen loss = 2.1605
Iteration 3200/9750: dis loss = 0.5001, gen loss = 2.6207
Iteration 3300/9750: dis loss = 0.6277, gen loss = 1.8626
Iteration 3400/9750: dis loss = 0.4362, gen loss = 1.6662
Iteration 3500/9750: dis loss = 0.5406, gen loss = 1.9908
```

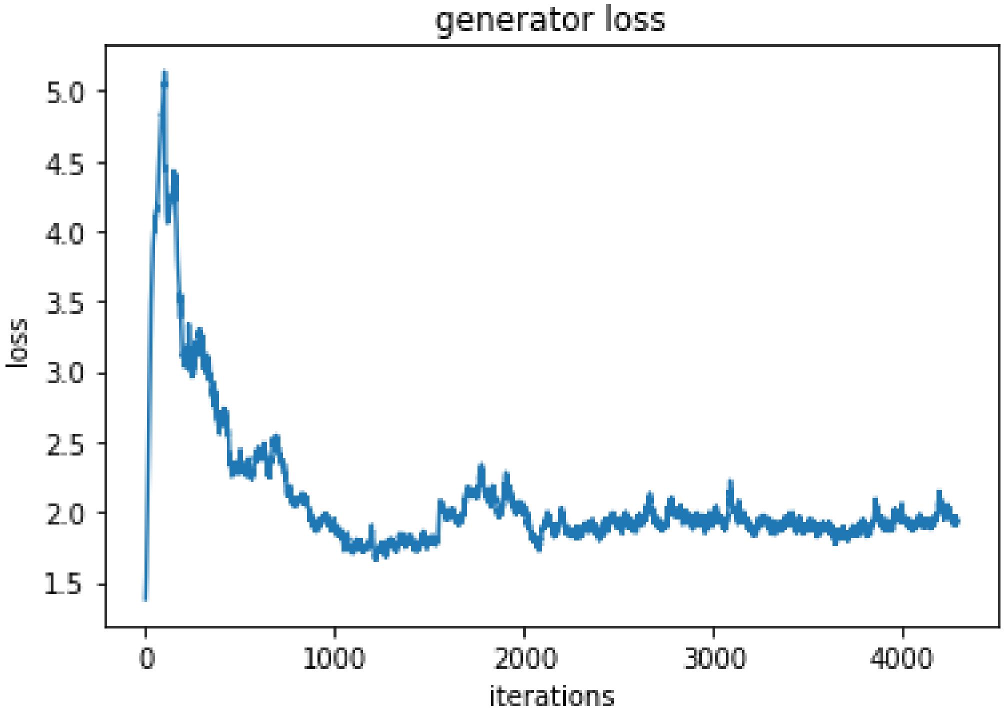

0
50
100
150
200
250
0
50
100
150
200
250



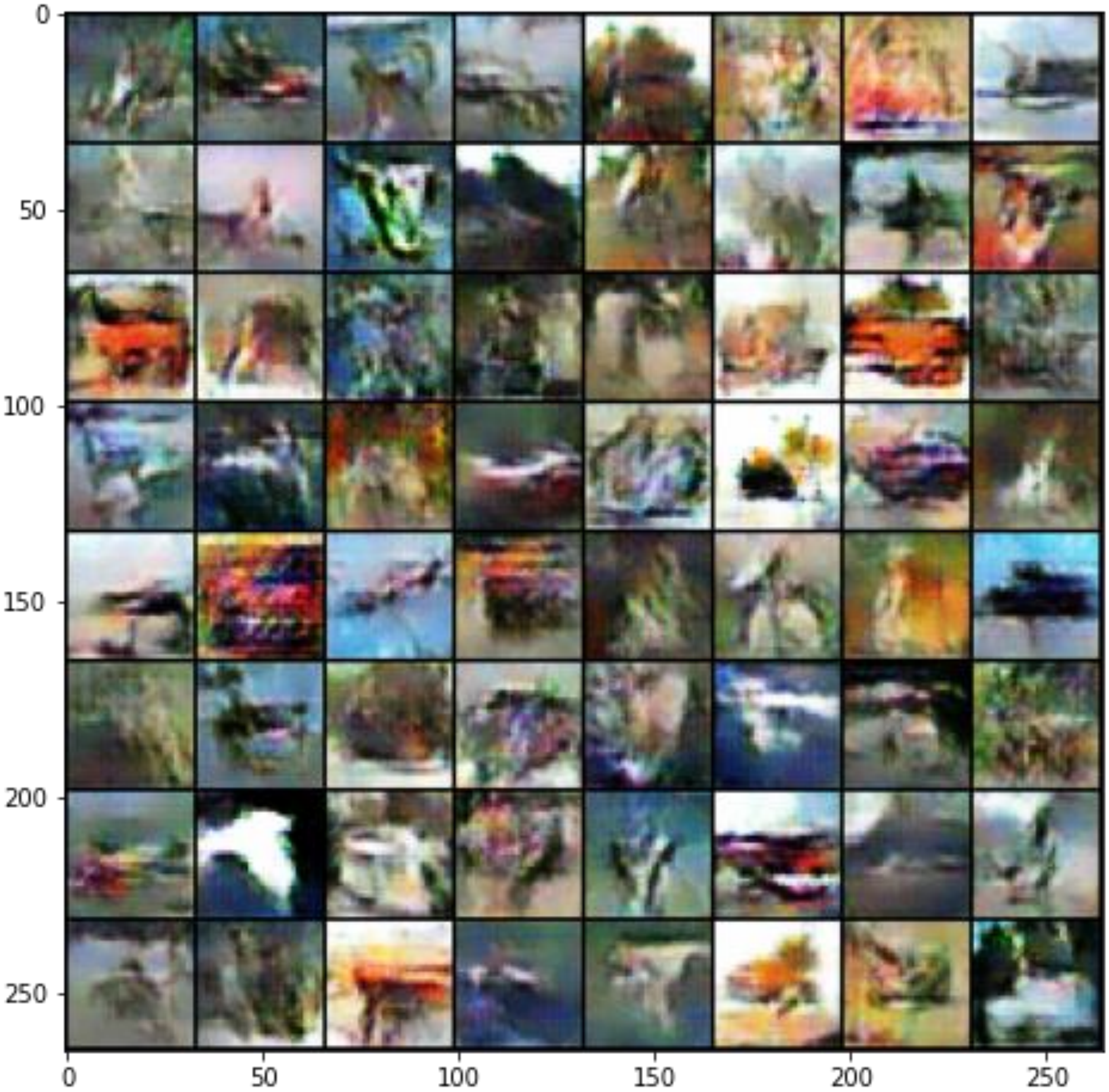
```
Iteration 3600/9750: dis loss = 0.4932, gen loss = 2.2087
Iteration 3700/9750: dis loss = 0.5014, gen loss = 1.9608
Iteration 3800/9750: dis loss = 0.5366, gen loss = 2.4596
Iteration 3900/9750: dis loss = 0.7695, gen loss = 2.7961
Iteration 4000/9750: dis loss = 0.7760, gen loss = 1.8016
Iteration 4100/9750: dis loss = 0.6765, gen loss = 1.8585
Iteration 4200/9750: dis loss = 0.6400, gen loss = 1.8719
```

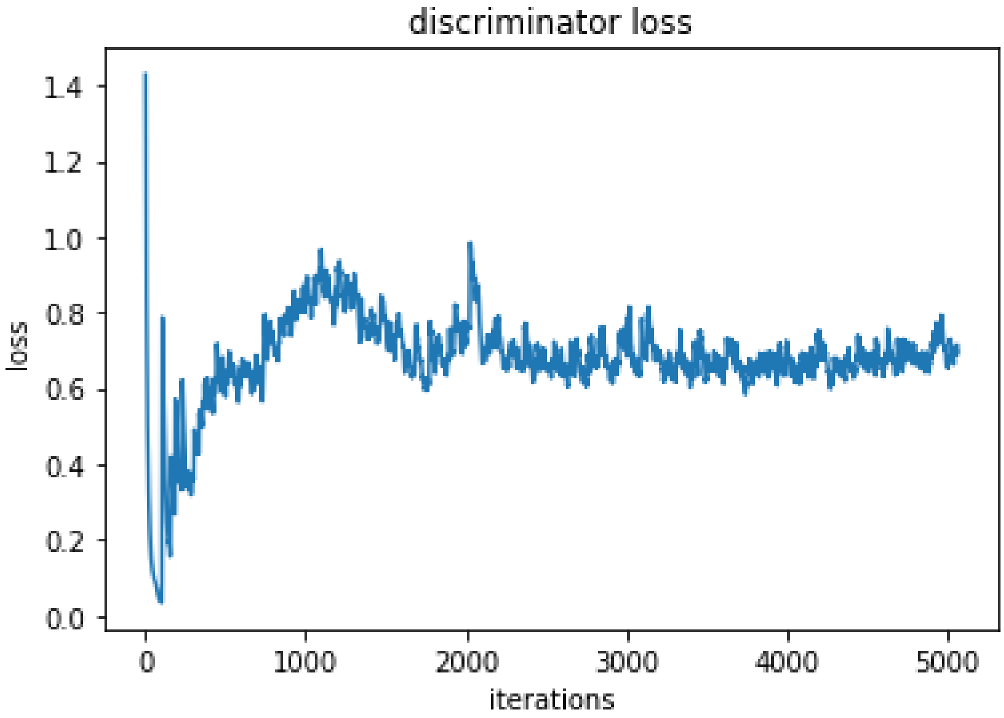

0
50
100
150
200
250
0
50
100
150
200
250

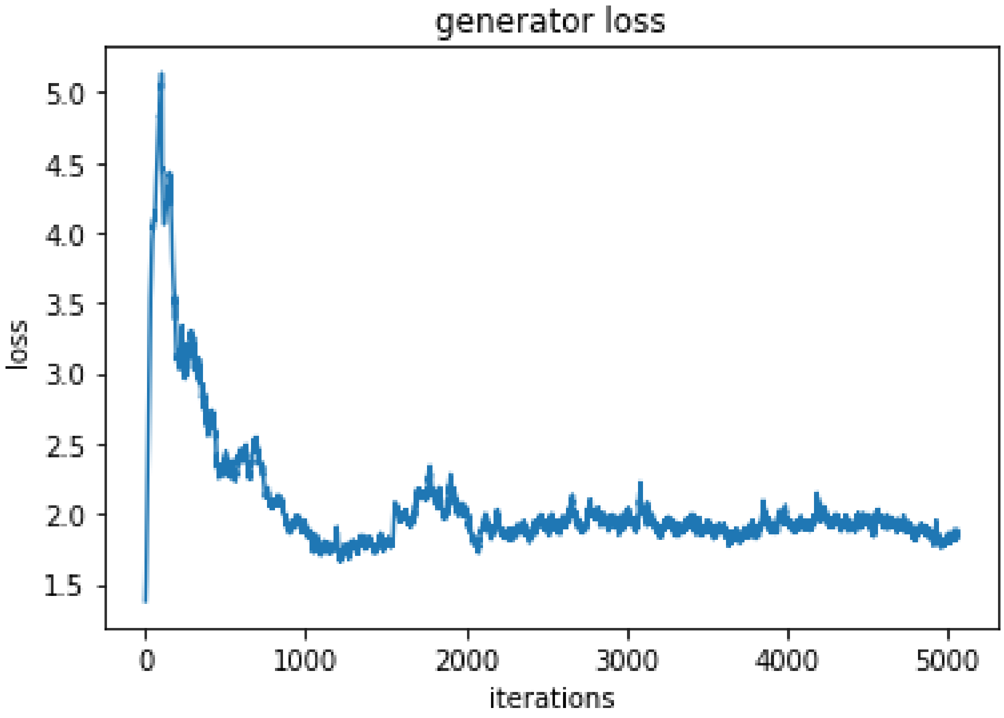




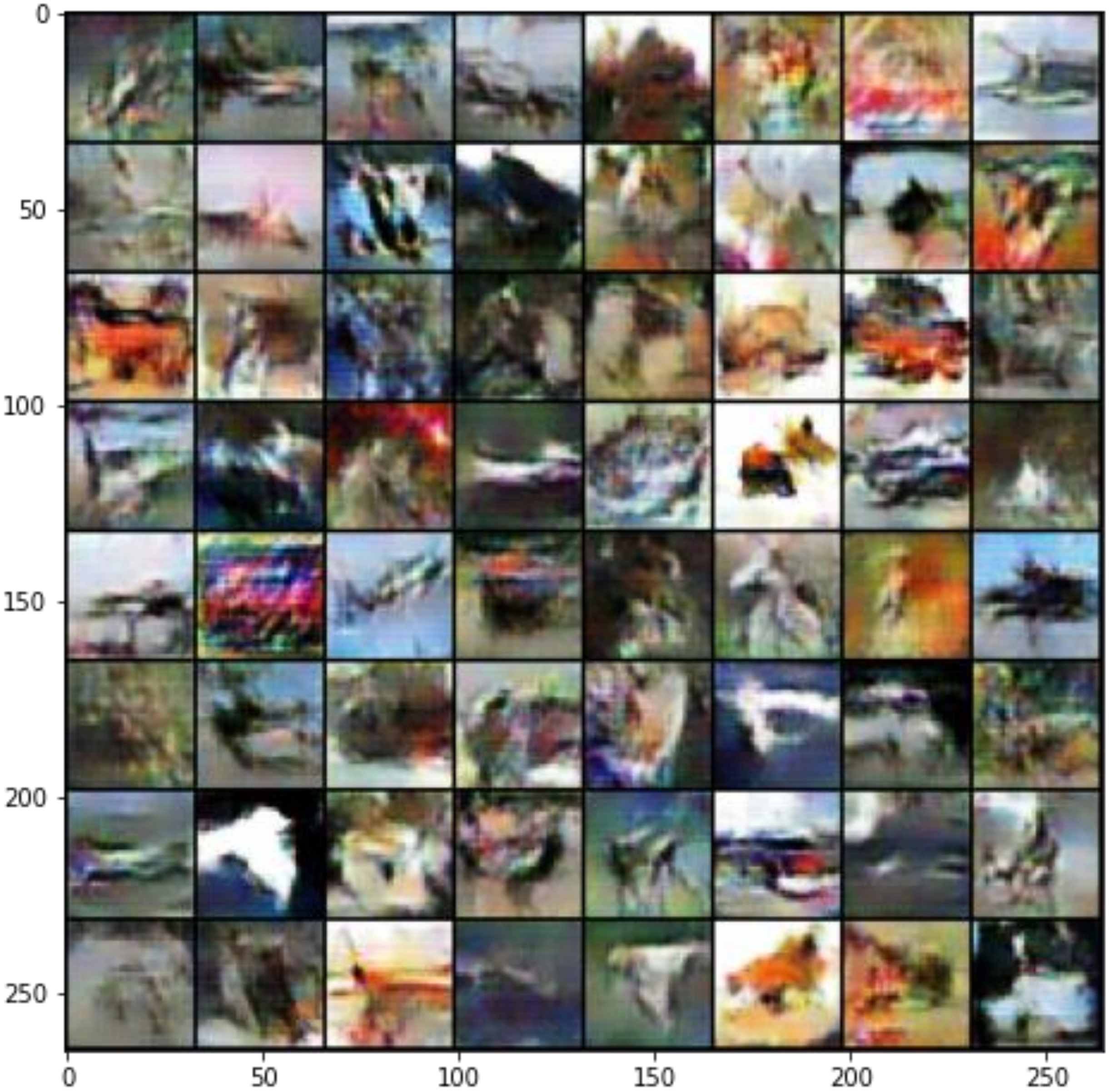
```
Iteration 4300/9750: dis loss = 0.7299, gen loss = 1.4466
Iteration 4400/9750: dis loss = 0.5617, gen loss = 2.0092
Iteration 4500/9750: dis loss = 0.7946, gen loss = 2.7517
Iteration 4600/9750: dis loss = 0.6602, gen loss = 1.1421
Iteration 4700/9750: dis loss = 0.7165, gen loss = 2.4949
Iteration 4800/9750: dis loss = 0.8666, gen loss = 2.0268
Iteration 4900/9750: dis loss = 0.7223, gen loss = 2.8066
Iteration 5000/9750: dis loss = 0.5812, gen loss = 1.4911
```

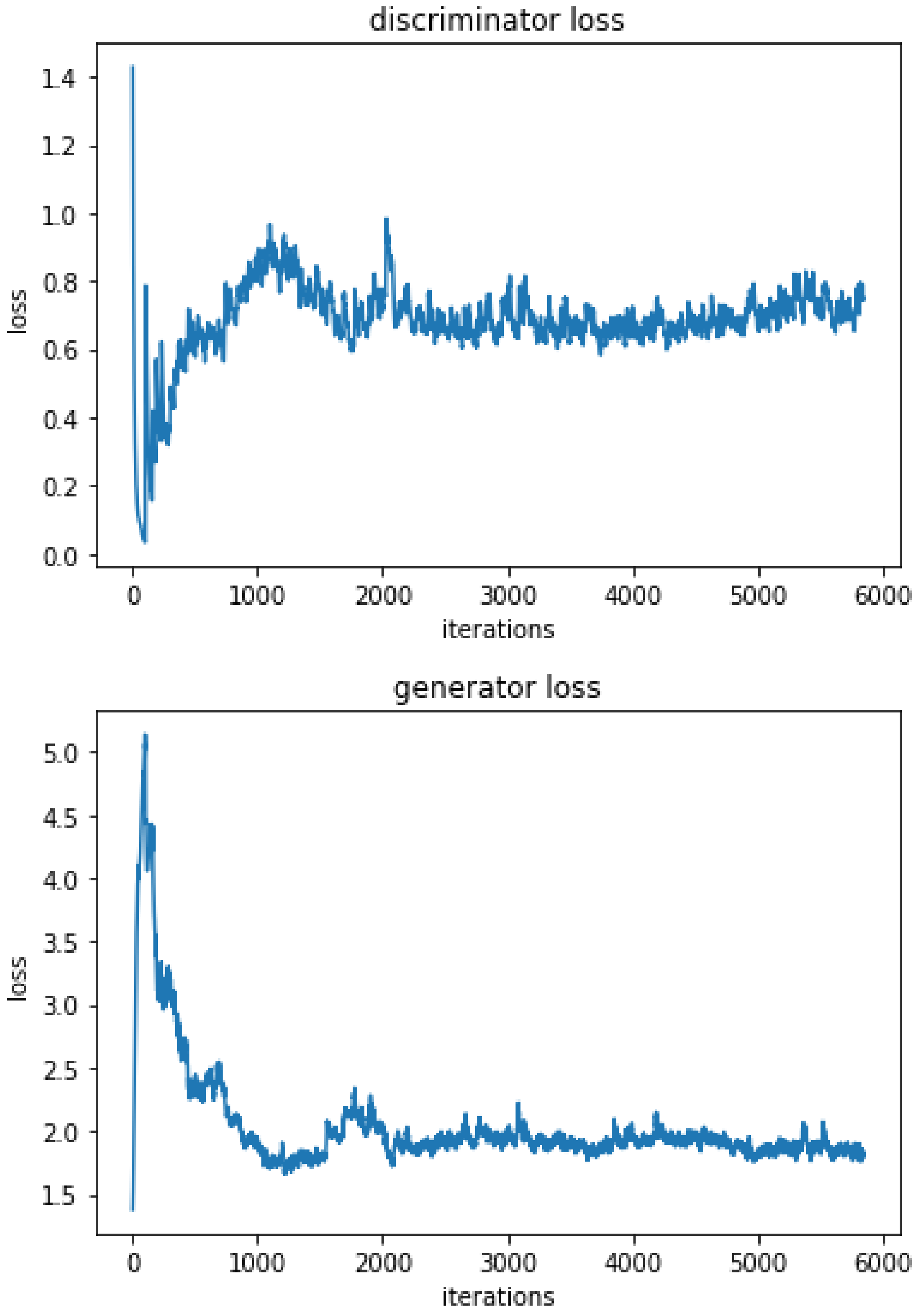

0
50
100
150
200
250
0
50
100
150
200
250



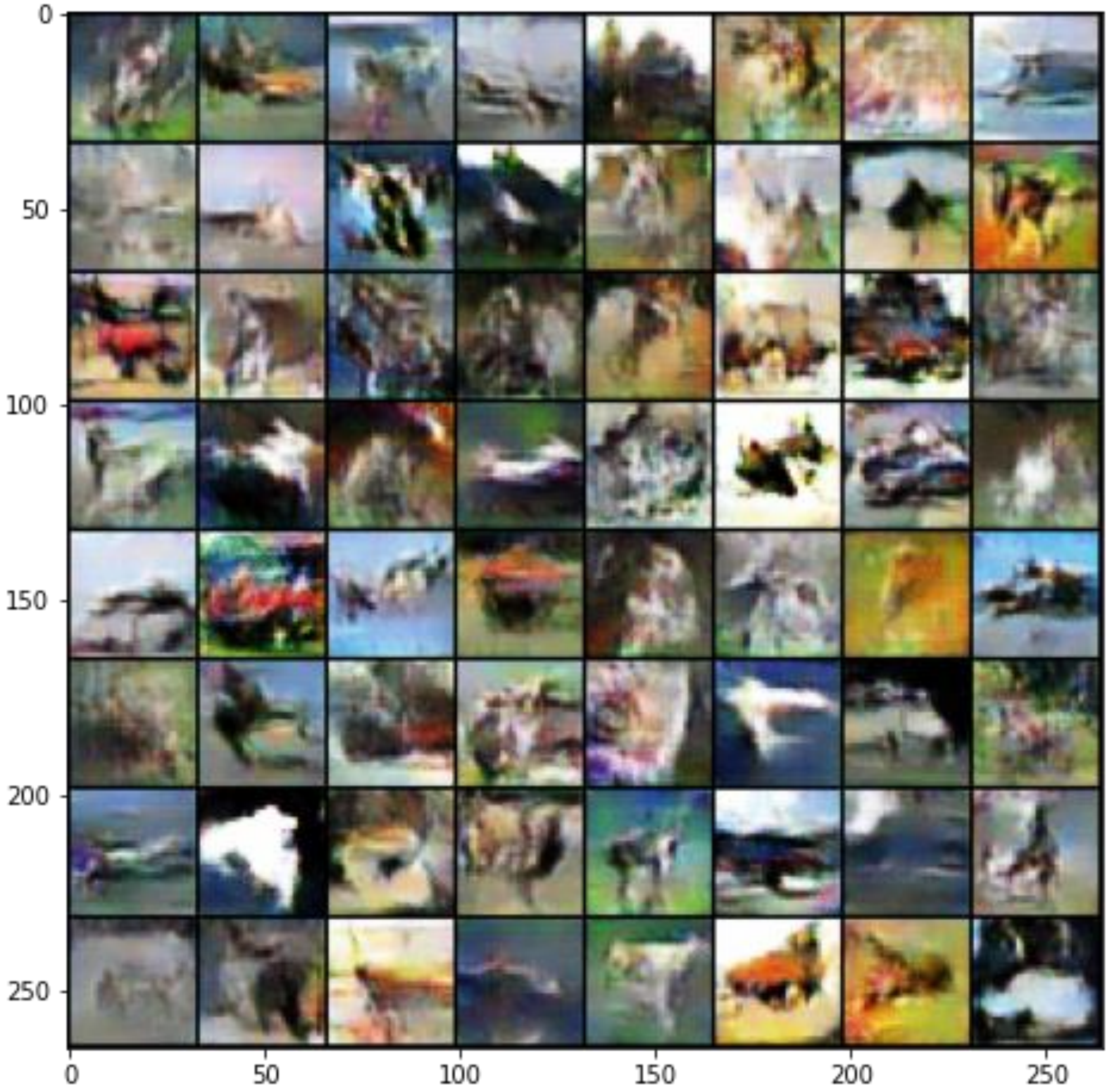


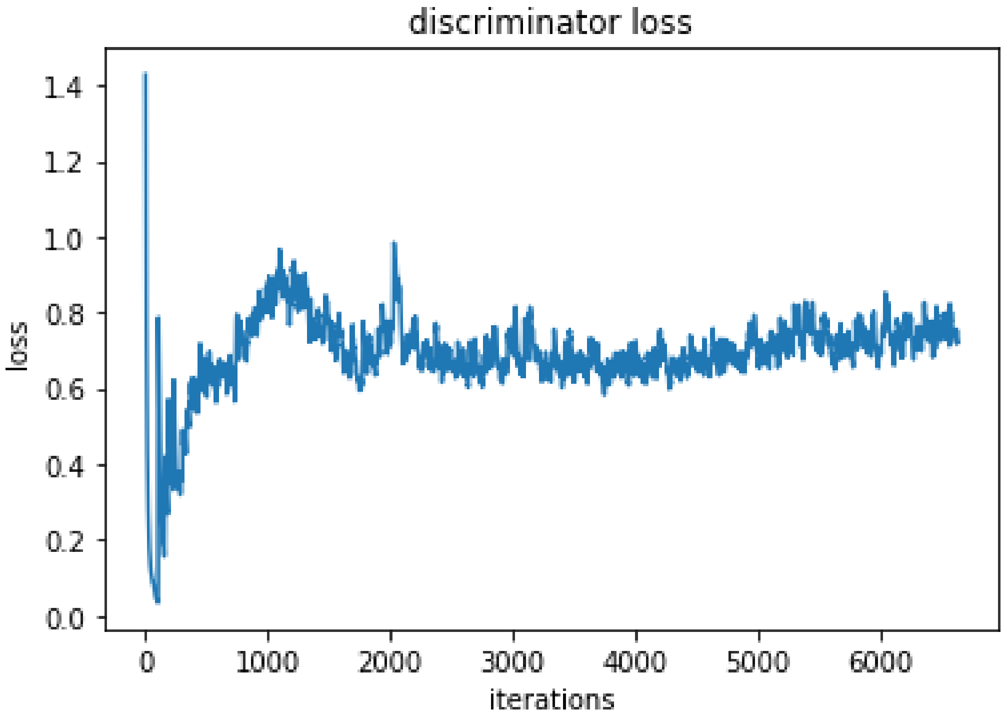
```
Iteration 5100/9750: dis loss = 1.4796, gen loss = 0.4255
Iteration 5200/9750: dis loss = 0.8290, gen loss = 2.8079
Iteration 5300/9750: dis loss = 0.7328, gen loss = 1.6509
Iteration 5400/9750: dis loss = 0.5847, gen loss = 2.0136
Iteration 5500/9750: dis loss = 0.7768, gen loss = 1.0723
Iteration 5600/9750: dis loss = 0.6497, gen loss = 2.3178
Iteration 5700/9750: dis loss = 0.7051, gen loss = 1.6641
Iteration 5800/9750: dis loss = 0.6031, gen loss = 1.8127
```

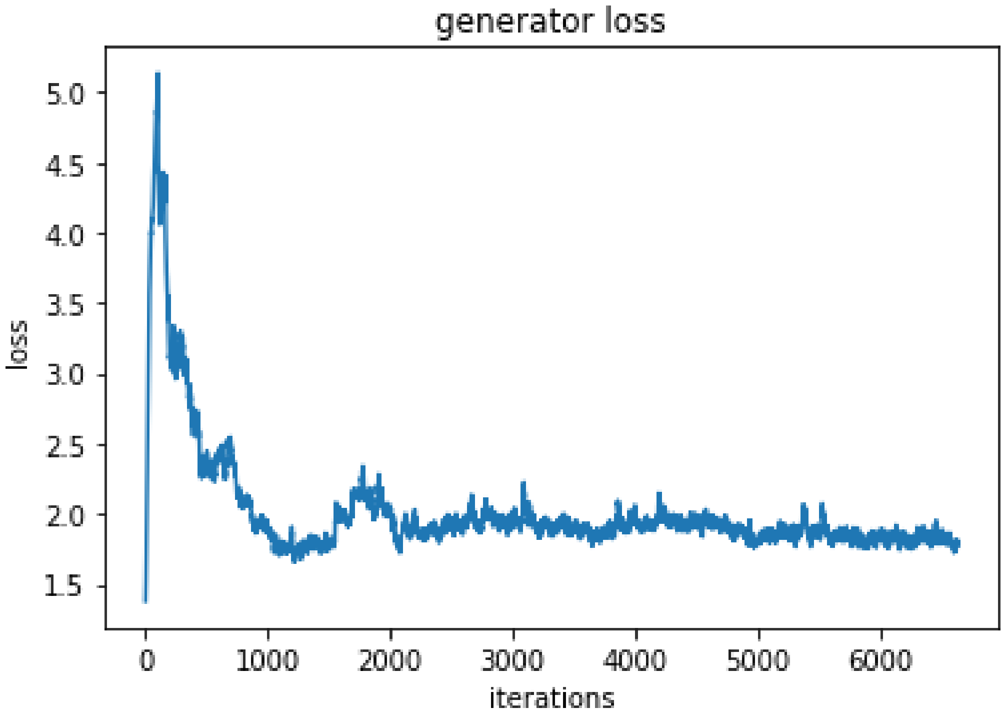

0
50
100
150
200
250
0
50
100
150
200
250



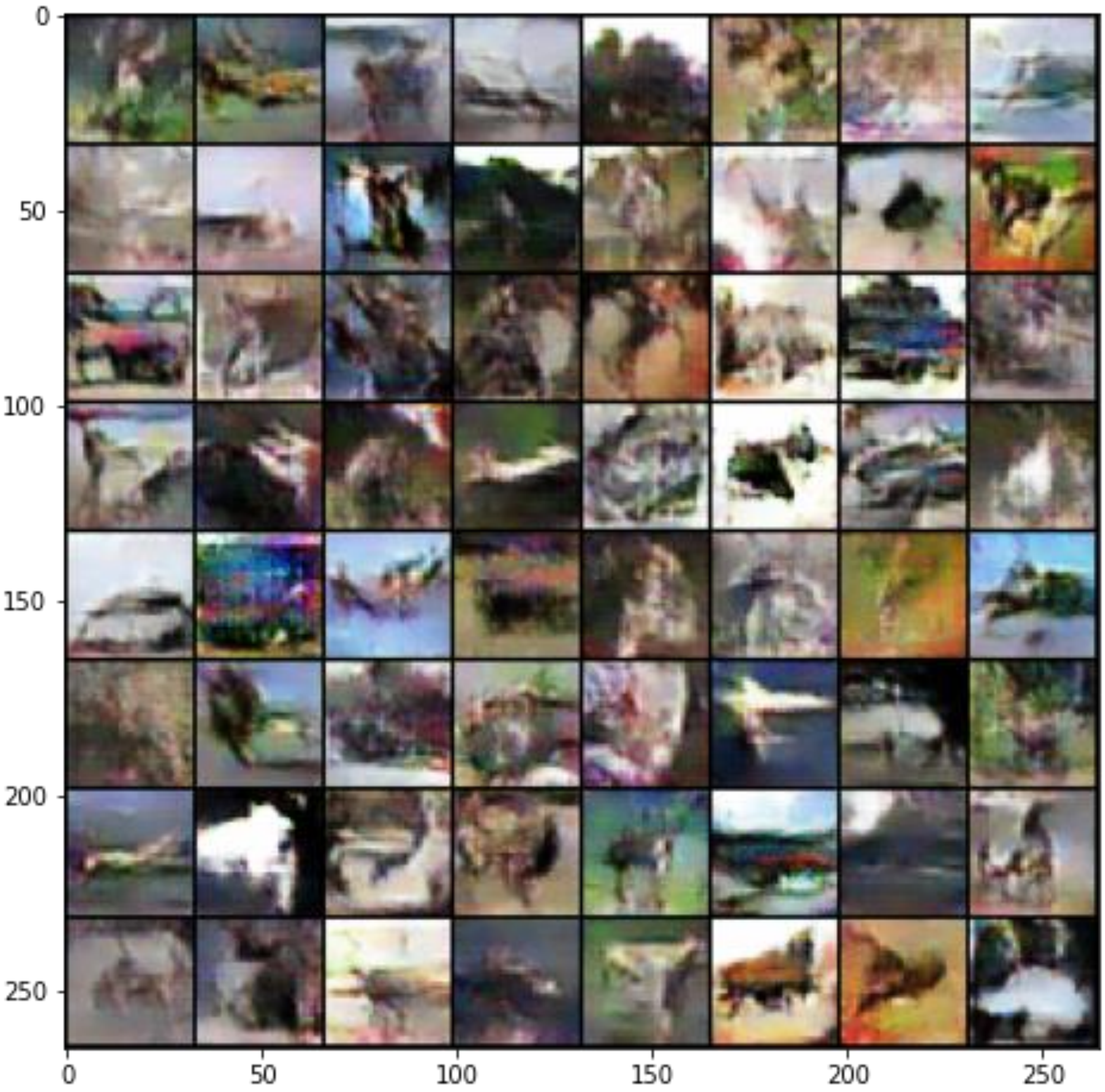
```
Iteration 5900/9750: dis loss = 0.6476, gen loss = 1.1970
Iteration 6000/9750: dis loss = 0.5720, gen loss = 2.6184
Iteration 6100/9750: dis loss = 0.5856, gen loss = 1.7594
Iteration 6200/9750: dis loss = 0.8030, gen loss = 0.7575
Iteration 6300/9750: dis loss = 0.5871, gen loss = 1.7265
Iteration 6400/9750: dis loss = 0.8192, gen loss = 1.5677
Iteration 6500/9750: dis loss = 0.9340, gen loss = 0.8170
Iteration 6600/9750: dis loss = 0.6816, gen loss = 1.7045
```

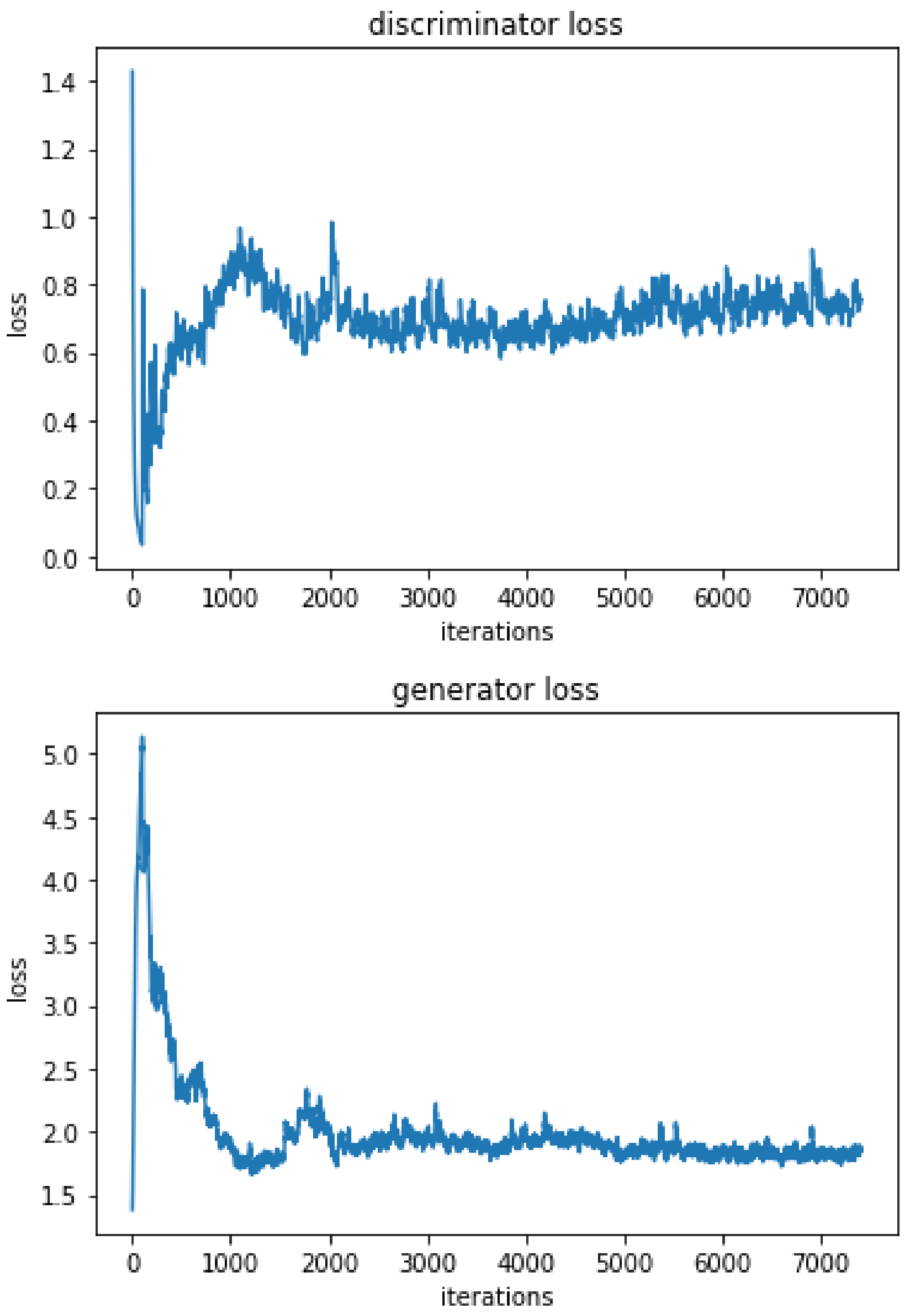

0
50
100
150
200
250
0
50
100
150
200
250



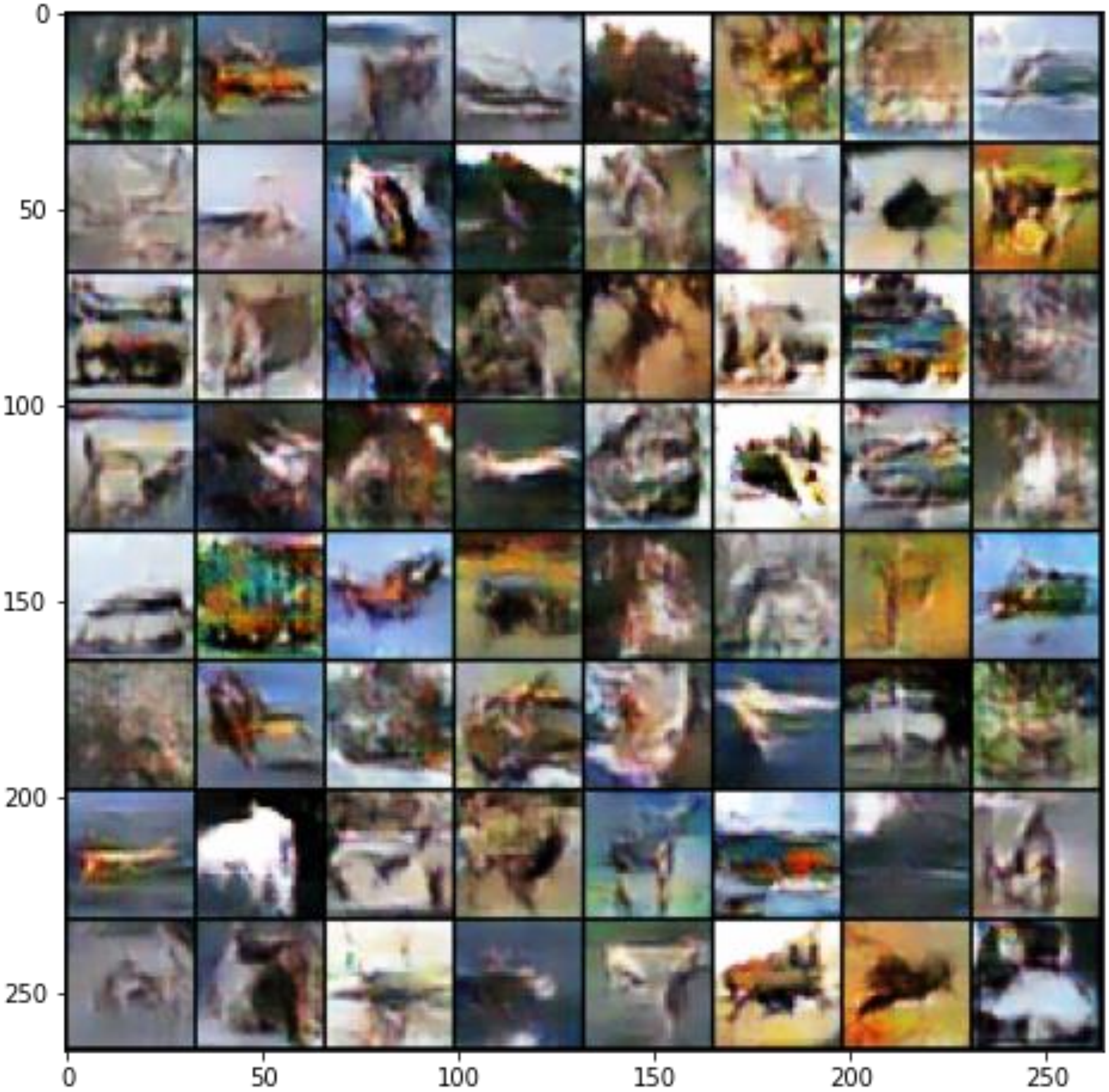


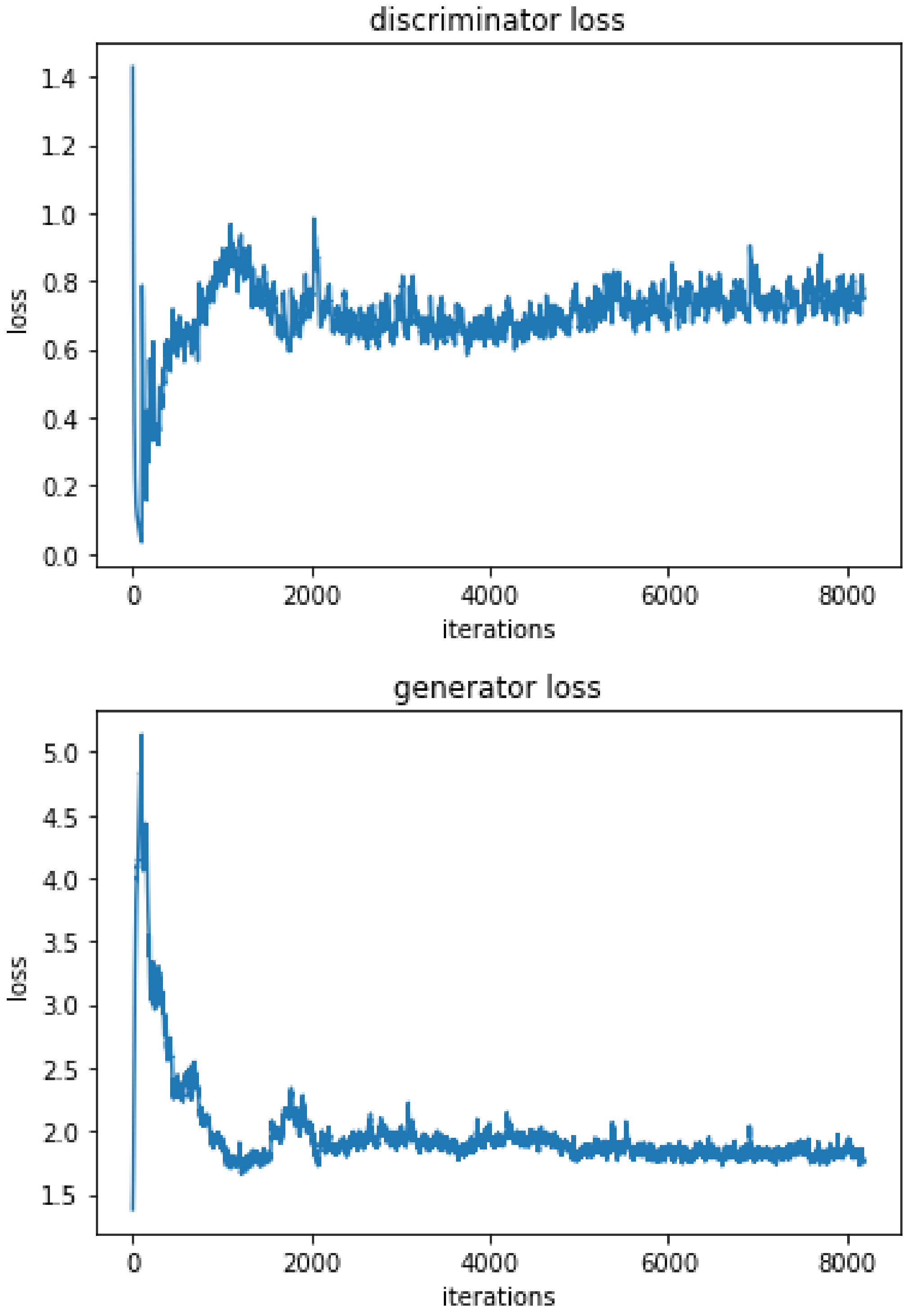
```
Iteration 6700/9750: dis loss = 0.7466, gen loss = 1.9352
Iteration 6800/9750: dis loss = 0.7541, gen loss = 1.9046
Iteration 6900/9750: dis loss = 0.7847, gen loss = 2.1983
Iteration 7000/9750: dis loss = 0.5317, gen loss = 1.8691
Iteration 7100/9750: dis loss = 0.8105, gen loss = 2.2554
Iteration 7200/9750: dis loss = 0.6283, gen loss = 1.7206
Iteration 7300/9750: dis loss = 0.6309, gen loss = 1.8506
Iteration 7400/9750: dis loss = 0.5781, gen loss = 2.2836
```


0
50
100
150
200
250
0
50
100
150
200
250

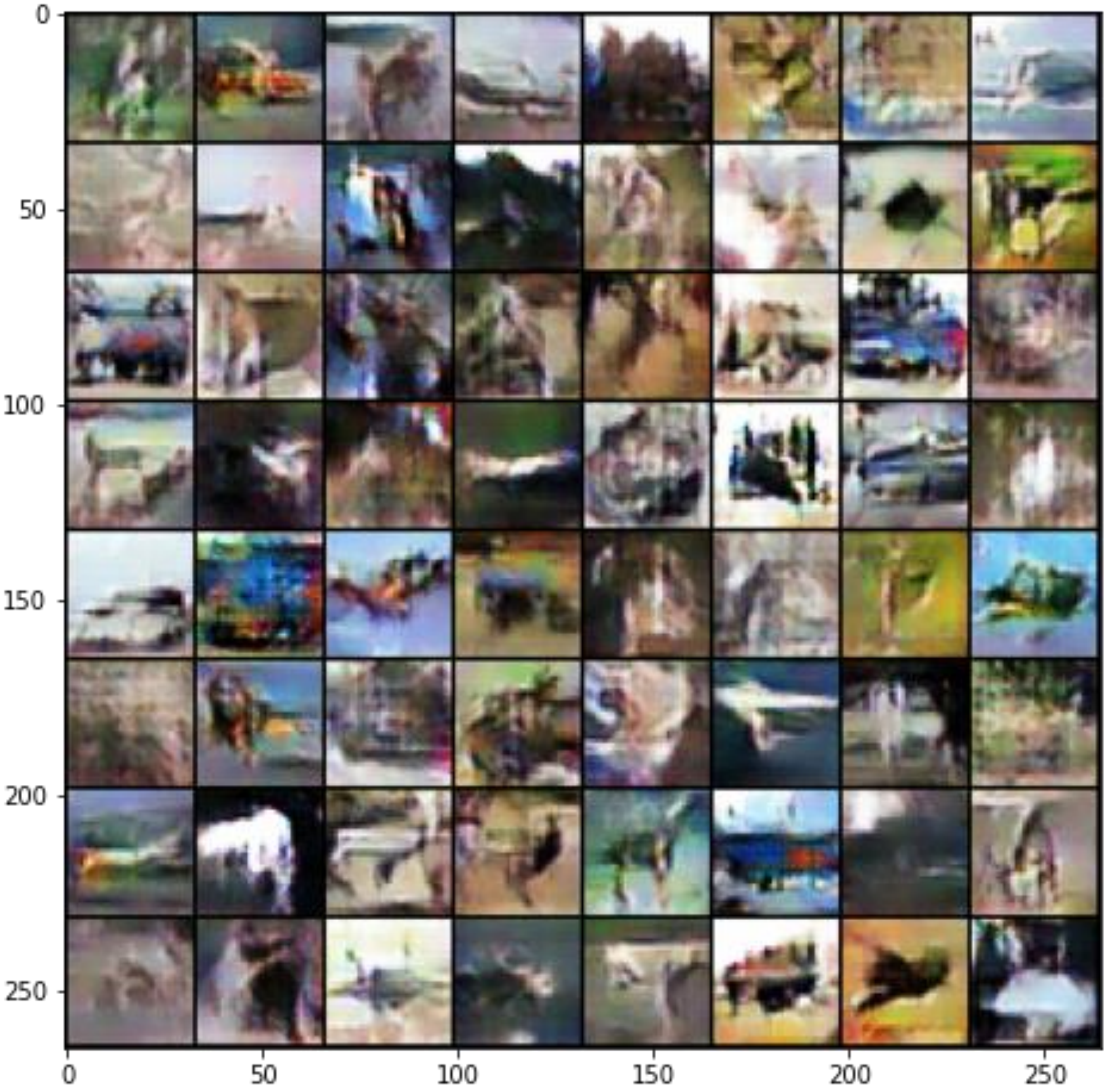


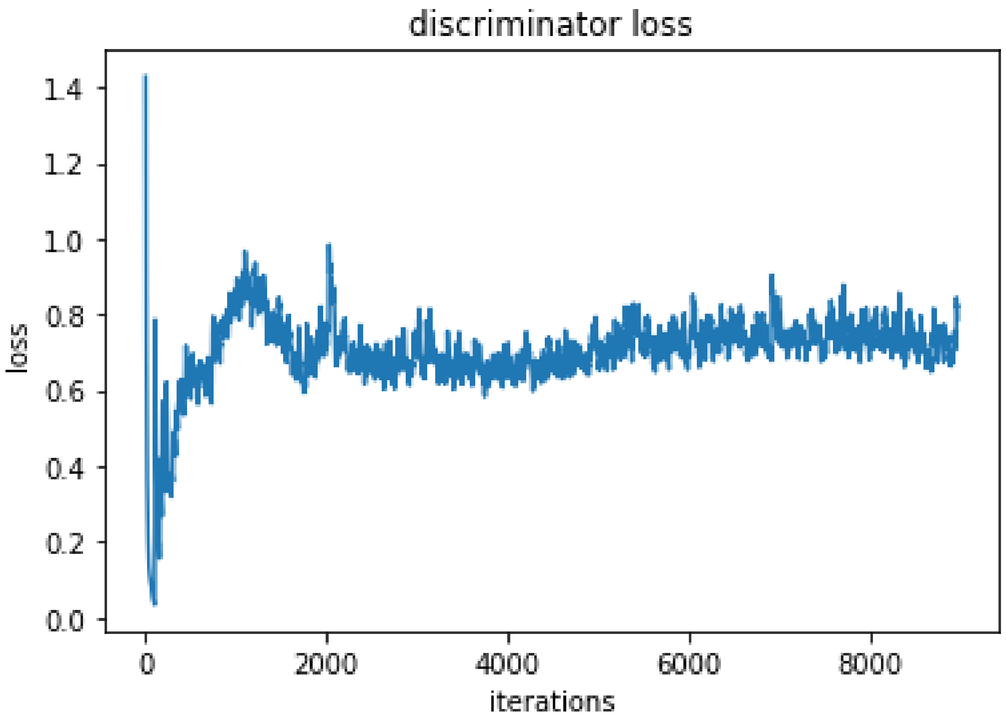
```
Iteration 7500/9750: dis loss = 0.7150, gen loss = 2.4038
Iteration 7600/9750: dis loss = 0.5901, gen loss = 1.9419
Iteration 7700/9750: dis loss = 0.7070, gen loss = 1.5974
Iteration 7800/9750: dis loss = 0.5520, gen loss = 1.9794
Iteration 7900/9750: dis loss = 0.7198, gen loss = 0.7561
Iteration 8000/9750: dis loss = 0.9980, gen loss = 0.9939
Iteration 8100/9750: dis loss = 0.5679, gen loss = 1.7823
```

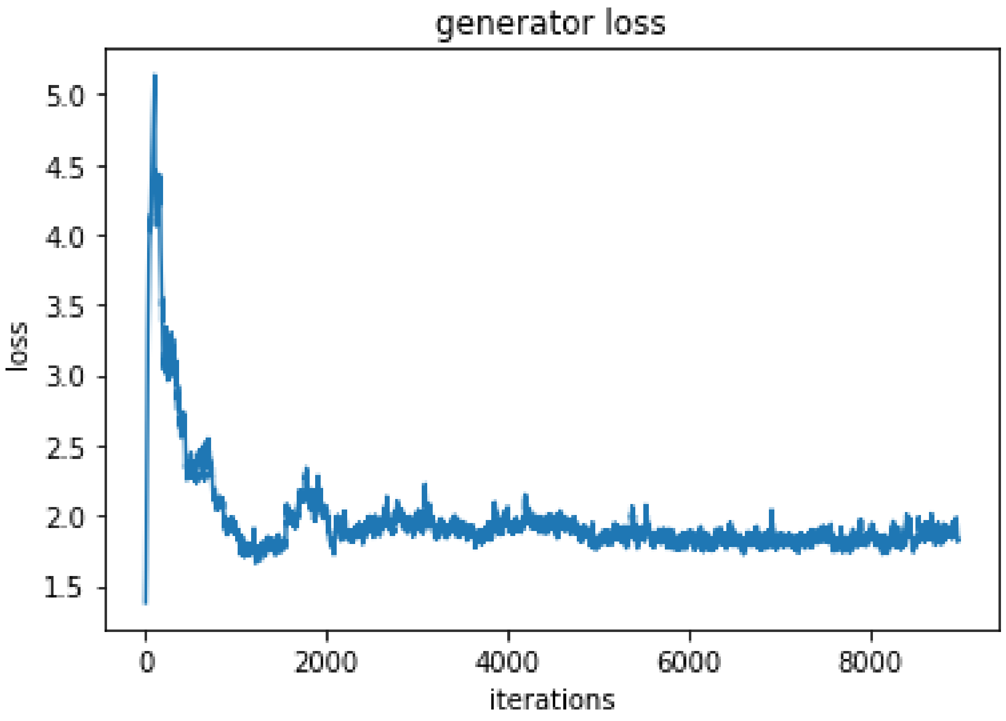

0
50
100
150
200
250
0
50
100
150
200
250



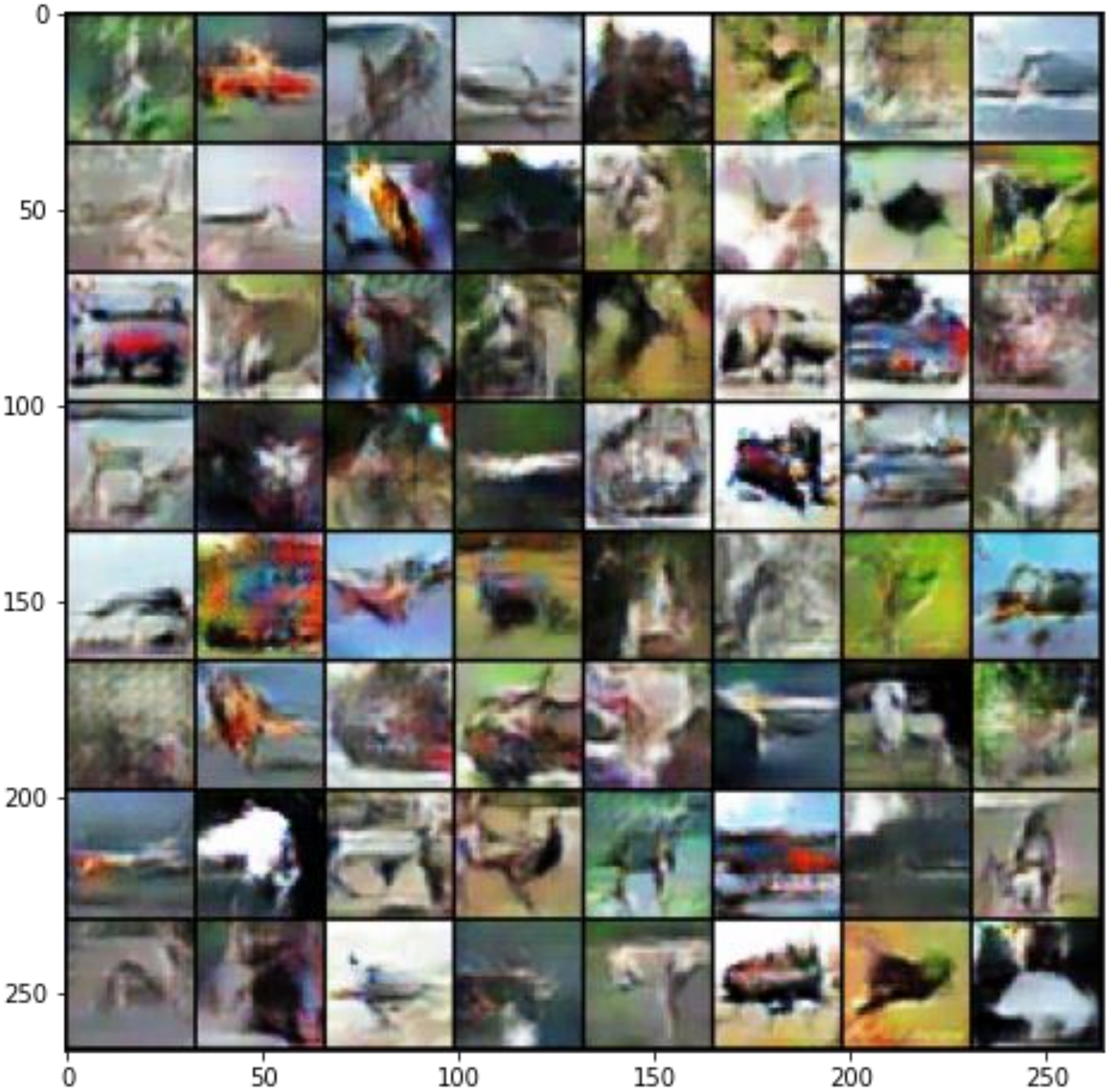
```
Iteration 8200/9750: dis loss = 0.6123, gen loss = 1.4255
Iteration 8300/9750: dis loss = 0.6078, gen loss = 1.3303
Iteration 8400/9750: dis loss = 0.6017, gen loss = 1.6096
Iteration 8500/9750: dis loss = 0.5962, gen loss = 2.6097
Iteration 8600/9750: dis loss = 0.6784, gen loss = 1.6843
Iteration 8700/9750: dis loss = 0.5767, gen loss = 2.2789
Iteration 8800/9750: dis loss = 0.6240, gen loss = 2.1209
Iteration 8900/9750: dis loss = 0.9147, gen loss = 2.6950
```

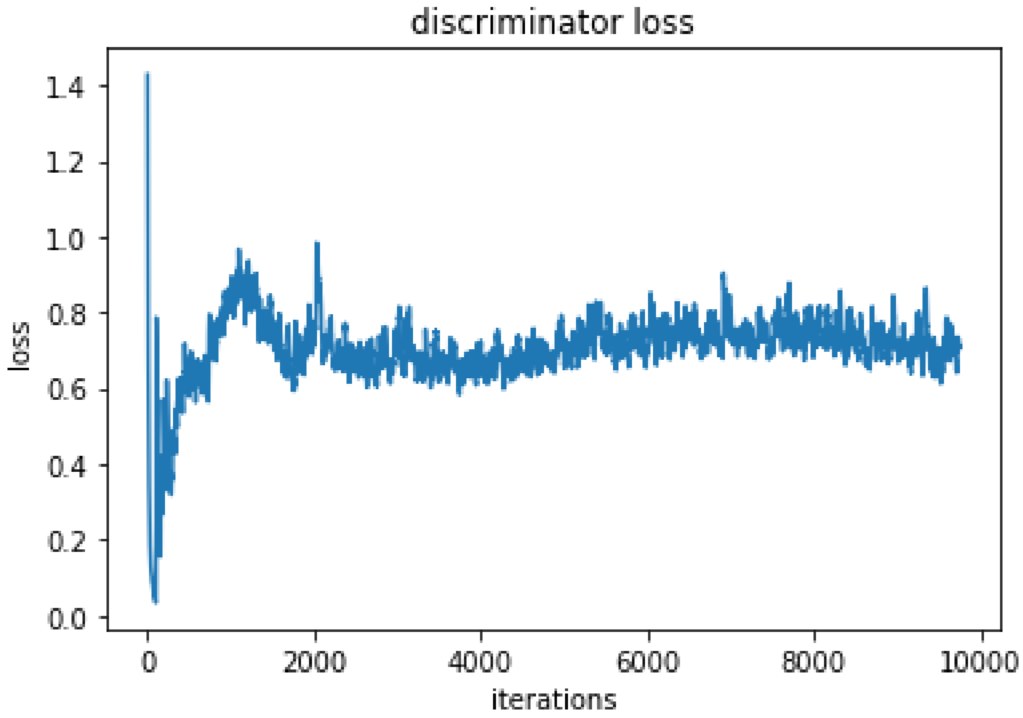

0
50
100
150
200
250
0
50
100
150
200
250

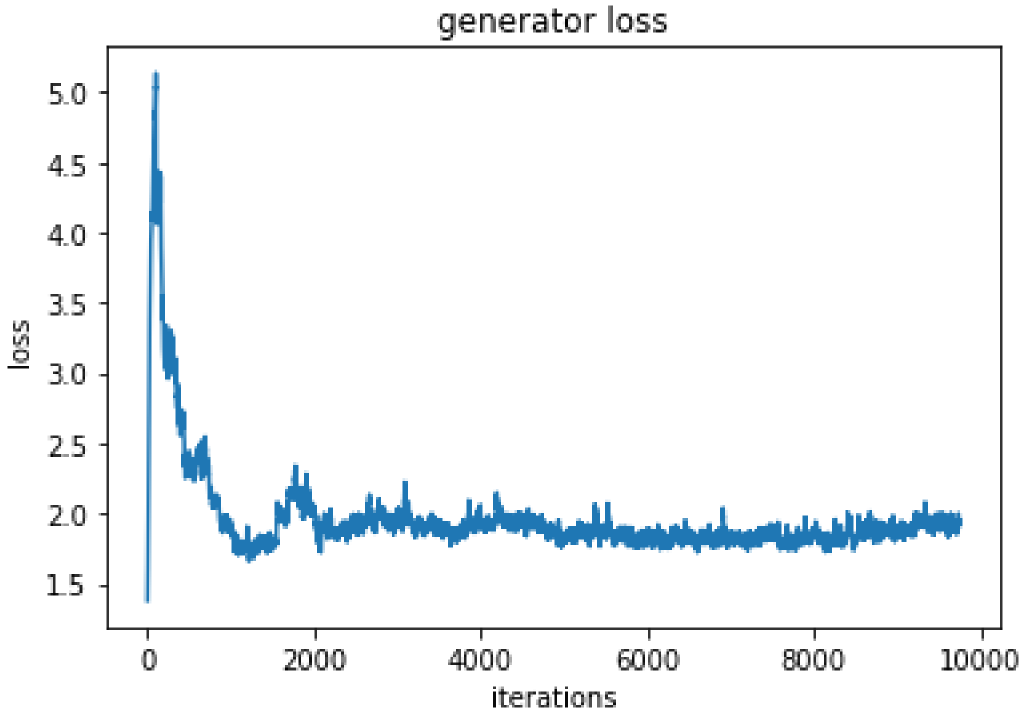




```
Iteration 9000/9750: dis loss = 0.8114, gen loss = 2.1705
Iteration 9100/9750: dis loss = 0.7722, gen loss = 1.4151
Iteration 9200/9750: dis loss = 0.6336, gen loss = 2.0441
Iteration 9300/9750: dis loss = 0.6362, gen loss = 1.8664
Iteration 9400/9750: dis loss = 0.6211, gen loss = 2.2766
Iteration 9500/9750: dis loss = 0.5894, gen loss = 1.4993
Iteration 9600/9750: dis loss = 0.5685, gen loss = 1.8596
Iteration 9700/9750: dis loss = 0.5196, gen loss = 1.7311
```


0
50
100
150
200
250
0
50
100
150
200
250


discriminator loss
loss
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000
iterations


generator loss
loss
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
0
2000
4000
6000
8000
10000
iterations

```
... Done!
```

Problem 2-2: The forger versus the police, revisited

Question: In the forger versus police story, we made part of it hand-wavy to hide a flaw that makes the story improbable to actually happen and makes it a bad analogy of how the training works in a GAN. Now that you have implemented a GAN, can you spot the flaw?

Specifically, when we consider one of the two parties, the other is treated as a black box. They know their opponent's result but not how they works. What is wrong here?

Answer: We can see that the generator has an indirect access to the discriminator’s parameters through the loss function, but the discriminator does not have access to the generator parameters. The generator is treated as a black box. The discriminator in the beginning stages of the training is better at knowing the fake samples, but it does not know how the generator works. Thus, the generator might be able to fool the discriminator but not in the optimal manner.

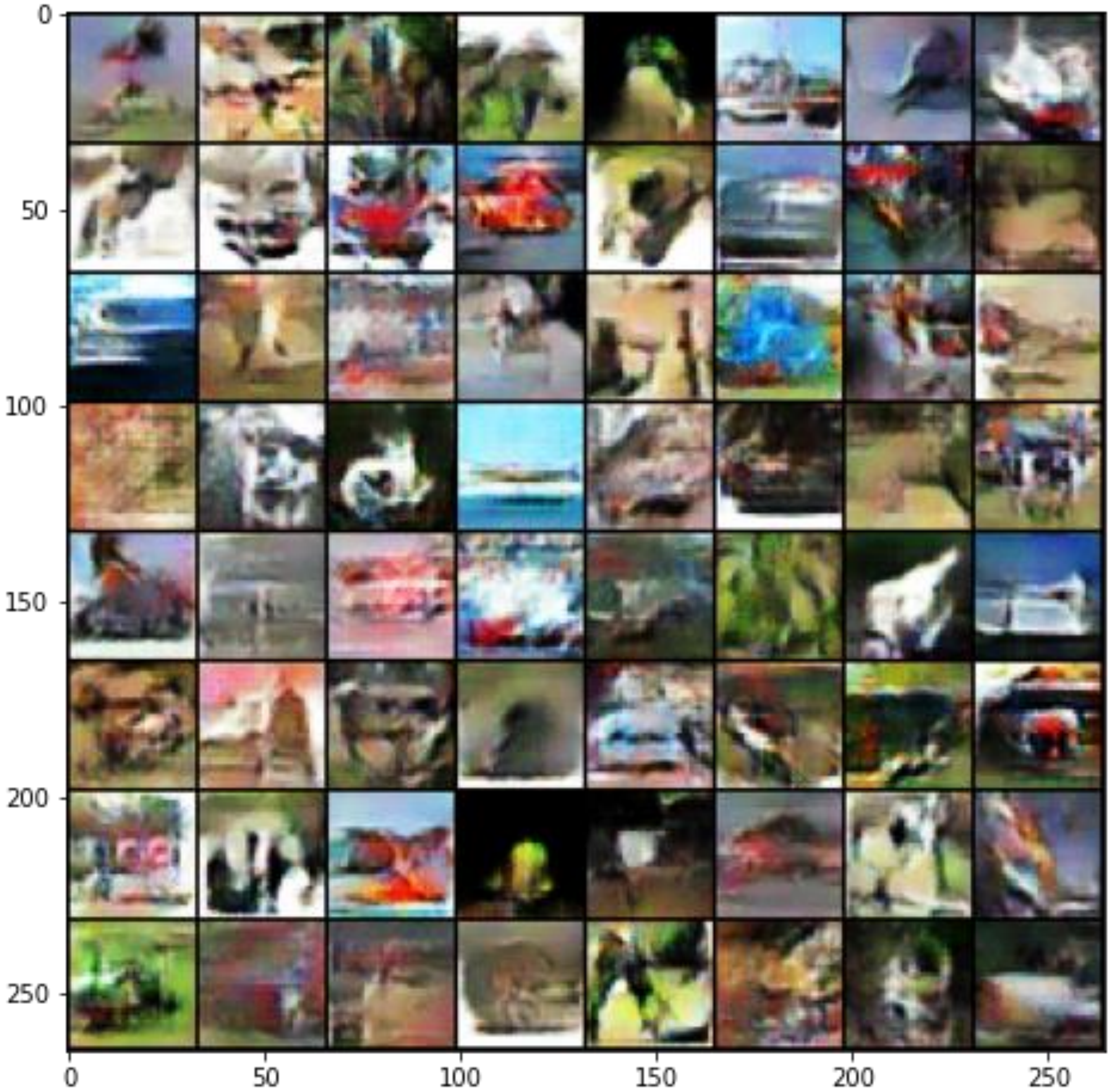
Problem 2-3: The Batch Normalization dilemma

Question: By removing the first batch normalization layer, for two different distributions to get confused with each other they must produce two distributions after dis_lrelu1 such that one can be obtained by applying an isotropic scaling and a translation to the other. Such a case is still possible but extremely unlikely to happen.

Propose a different way of feeding the samples to solve the problem in the second question without omitting any batch normalization layers or changing their mode of operation.

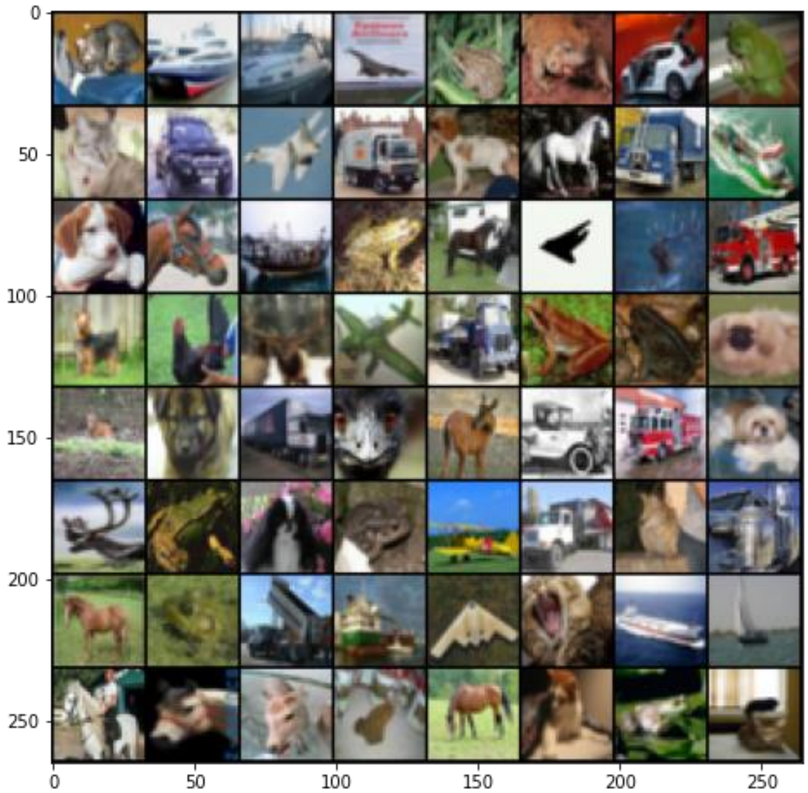
Answer, The real and fake samples should be made very different before passing into the discriminator. Both the real samples and fake samples can undergo a nonlinear transformation before passing it to the discriminator such that the one sample cannot be obtained by applying isotropic scaling and a translation to the other. Thus, the batch normalization layer would not affect the model, and the two different distributions would not get confused with each other.

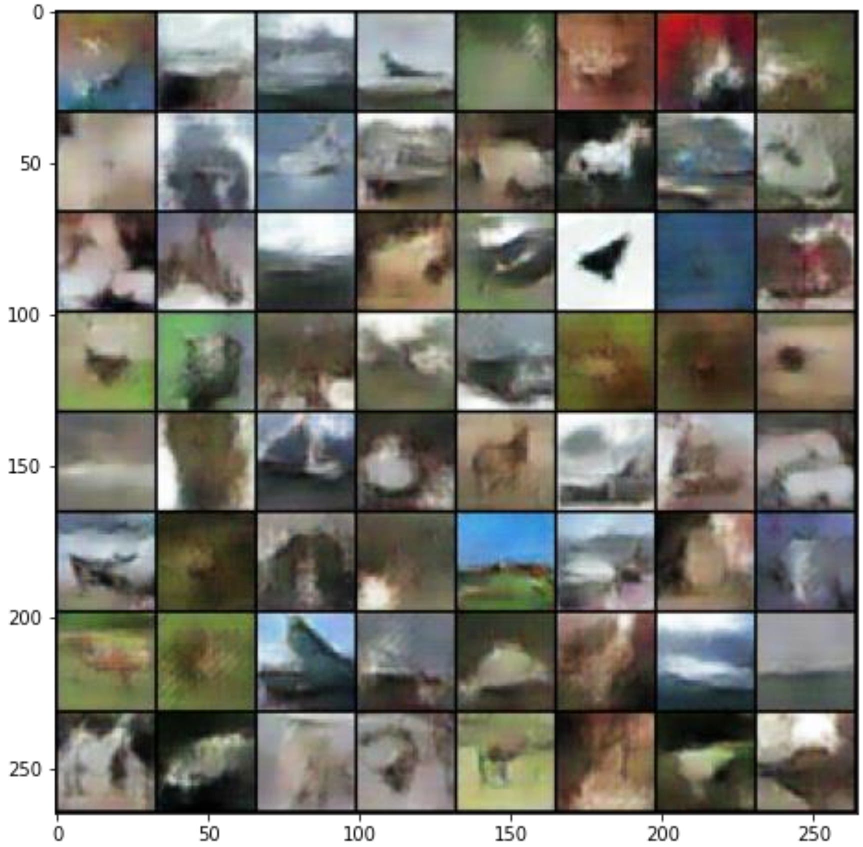
Activation Maximization



For this part, you need to achieve a reconstruction loss < 0.0145. Do NOT modify anything outside of the blocks marked for you to fill in.

```
average reconstruction loss = 0.0133
```


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250