COMP4680/COMP8650: Advanced Topics in SML

Assignment #6: Deep Learning Programming Assignment

ID: u6261174

Email：u6261174@anu.edu.au

1.

a) The size of the output: (32, 16, 64, 64) 32 batch sizes16 channels image 64*64

b) The size of the output: (32, 128, 3, 3) 32 batch sizes 128 channels image 3*3

c) The size of the output: (32, 32, 65, 65) 32 batch sizes 32 channels image 65*65

d) The number of learnable parameters: 448 (16*3*3+16)

2.

a) The source code show as below:

```
def __init__(self, z_dim=64, img_size=64):
    super(Encoder, self).__init__()

    self.z_dim = z_dim
    self.hidden_layer = nn.Sequential(
        nn.Linear(3*img_size*img_size, z_dim),
        nn.BatchNorm1d(z_dim),
        nn.ReLU(),
    )
    self.output_layer = nn.Sequential(
        nn.Linear(z_dim, z_dim),
        nn.Tanh(),
    )
```

b) The source code show as below:

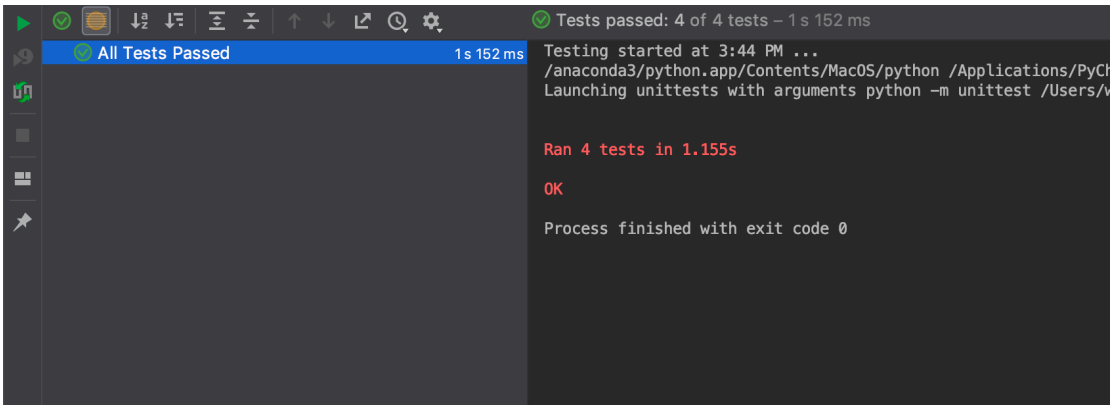
```
def __init__(self, z_dim=64, img_size=64):
    super(Decoder, self).__init__()

    assert img_size==64
    self.z_dim = z_dim
    self.layer1 = nn.Sequential(
        nn.ConvTranspose2d(
            in_channels=z_dim,
            out_channels=128,
            kernel_size=7,
            stride=1,
        ),
        nn.BatchNorm2d(128),
        nn.ReLU(),
        nn.Conv2d(
            in_channels=128,
            out_channels=128,
            kernel_size=3,
            stride=1,
            padding=1,
        ),
        nn.BatchNorm2d(128),
        nn.ReLU(),
    )
    self.layer2 = nn.Sequential(
        nn.ConvTranspose2d(
            in_channels=128,
            out_channels=64,
            kernel_size=3,
            stride=2,
        ),
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.Conv2d(
            in_channels=64,
            out_channels=64,
            kernel_size=3,
            stride=1,
            padding=1,
        ),
        nn.BatchNorm2d(64),
        nn.ReLU(),
    )
```

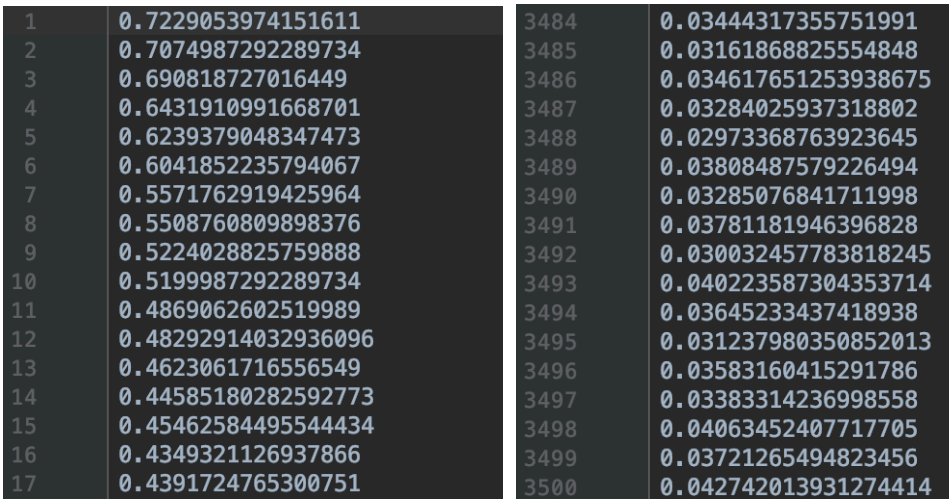
```
self.layer3 = nn.Sequential(
    nn.ConvTranspose2d(
        in_channels=64,
        out_channels=64,
        kernel_size=3,
        stride=2,
    ),
    nn.BatchNorm2d(64),
    nn.ReLU(),
    nn.Conv2d(
        in_channels=64,
        out_channels=64,
        kernel_size=3,
        stride=1,
        padding=1,
    ),
    nn.BatchNorm2d(64),
    nn.ReLU(),
)
self.layer4 = nn.Sequential(
    nn.ConvTranspose2d(
        in_channels=64,
        out_channels=32,
        kernel_size=3,
        stride=2,
    ),
    nn.BatchNorm2d(32),
    nn.ReLU(),
    nn.Conv2d(
        in_channels=32,
        out_channels=32,
        kernel_size=3,
        stride=1,
        padding=1,
    ),
    nn.BatchNorm2d(32),
    nn.ReLU(),
)
self.layer5 = nn.Sequential(
    nn.ConvTranspose2d(
        in_channels=32,
        out_channels=32,
        kernel_size=2,
        stride=1,
    ),
    nn.BatchNorm2d(32),
    nn.ReLU(),
    nn.Conv2d(
        in_channels=32,
        out_channels=3,
        kernel_size=1,
        stride=1,
        padding=0,
    ),
    nn.Tanh(),
)
```

3.

a) All the models layers are correctly sized. The check results show as below:



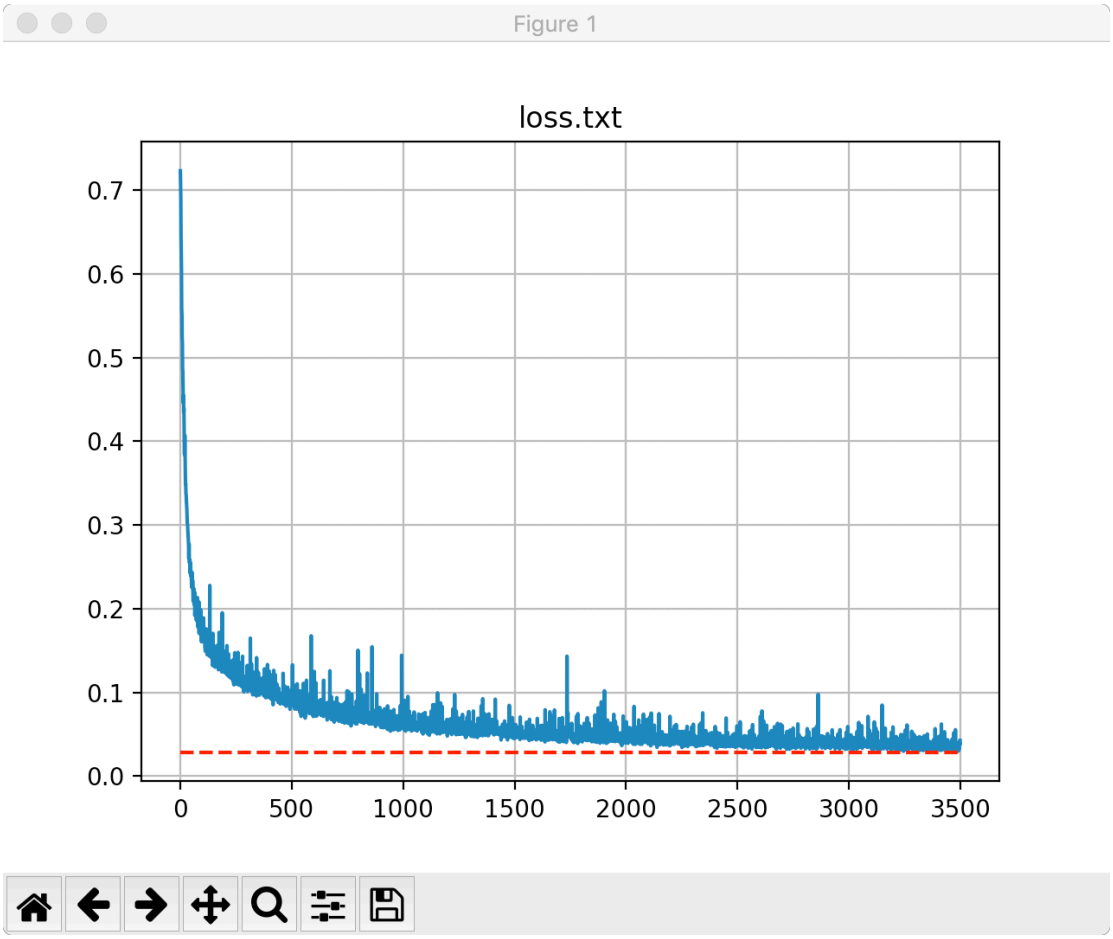
b) The results are stored in the loss.txt after traning. The following images shows a part of results(loss.txt).



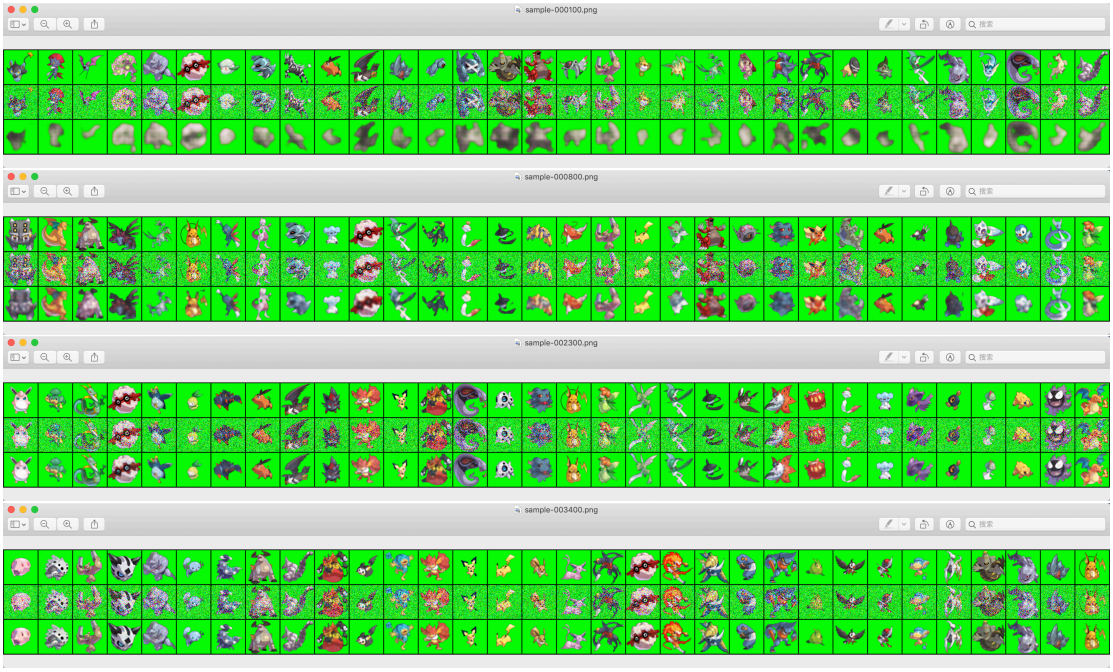
1	0.7229053974151611
2	0.7074987292289734
3	0.690818727016449
4	0.6431910991668701
5	0.6239379048347473
6	0.6041852235794067
7	0.5571762919425964
8	0.5508760809898376
9	0.5224028825759888
10	0.5199987292289734
11	0.4869062602519989
12	0.48292914032936096
13	0.4623061716556549
14	0.44585180282592773
15	0.45462584495544434
16	0.4349321126937866
17	0.4391724765300751

3484	0.03444317355751991
3485	0.03161868825554848
3486	0.034617651253938675
3487	0.03284025937318802
3488	0.02973368763923645
3489	0.03808487579226494
3490	0.03285076841711998
3491	0.03781181946396828
3492	0.030032457783818245
3493	0.040223587304353714
3494	0.03645233437418938
3495	0.031237980350852013
3496	0.03583160415291786
3497	0.03383314236998558
3498	0.04063452407717705
3499	0.03721265494823456
3500	0.042742013931274414

c) The loss function show as below. According to the following image, the loss function seems like converged.

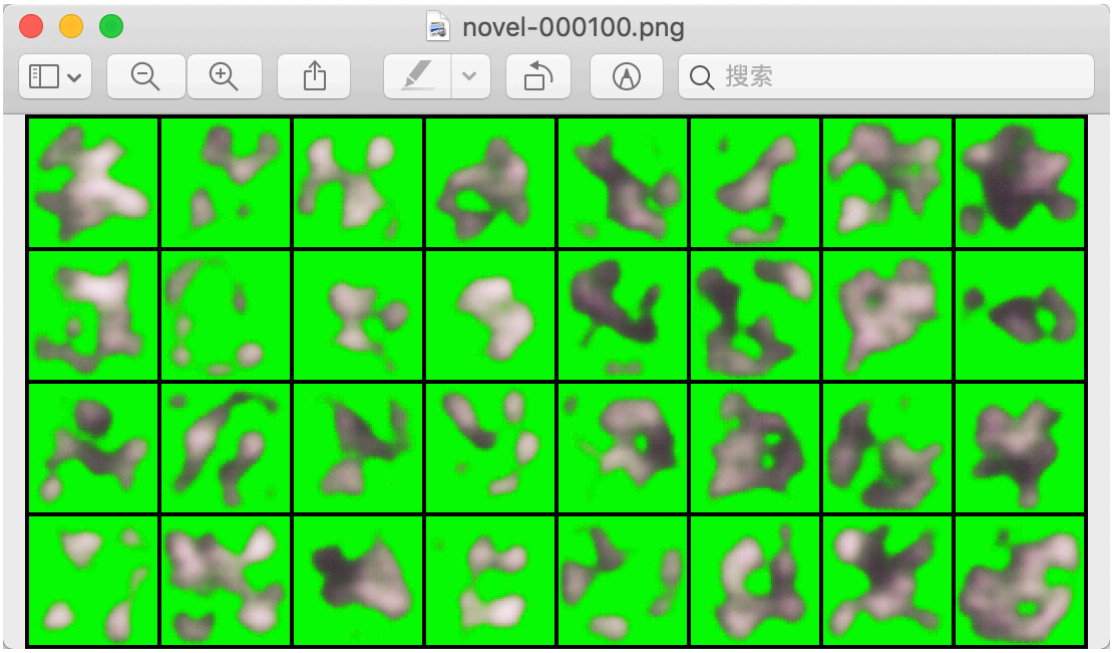


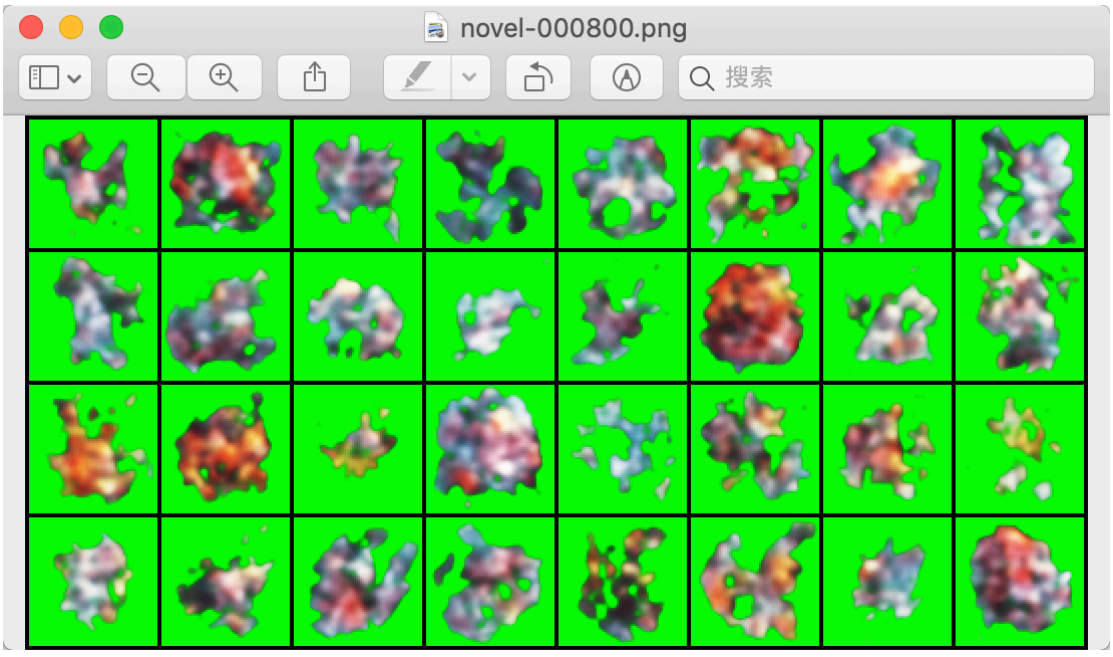
d) According to the images of sample-*.png, the quality of the denoised images become higher as the training processs. Therefore, this can deal with this situation. The following images are the results after 100 iterations, 800 iterations, 2300 iterations, 3400 iterations respectively.

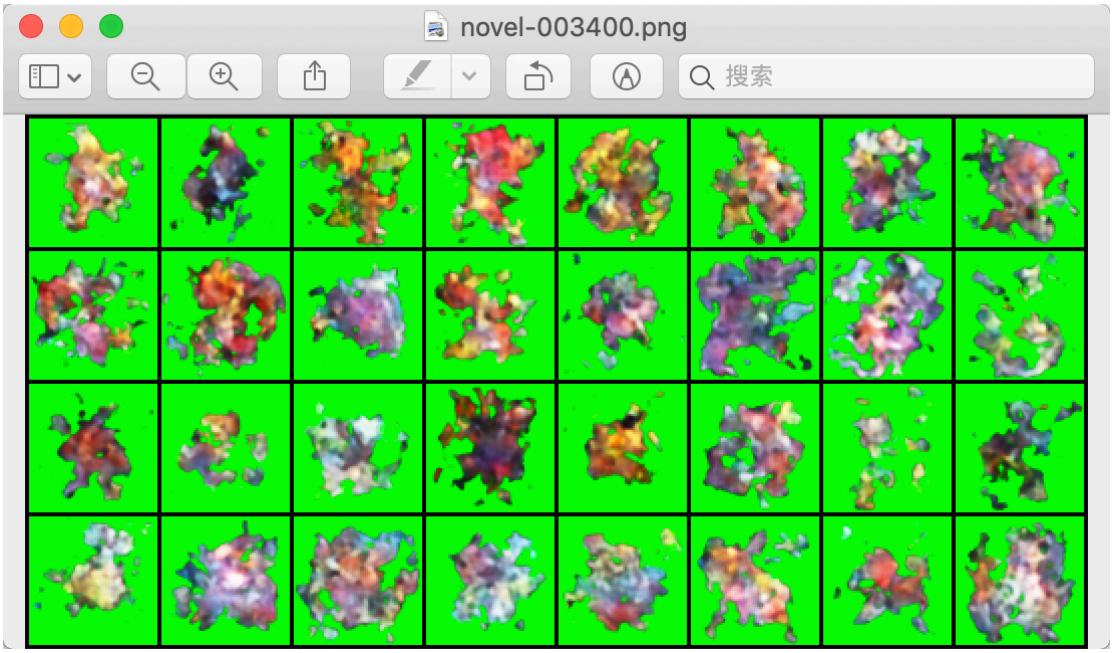


e) According to the images of novel-*.png, the quality of the images generated from

random noise injected into the decoder model become higher. But the final results are still hard to recognize. Therefore, this decoder model can not deal with this situation. The following images are the results after 100 iterations, 800 iterations, 2300 iterations, 3400 iterations respectively.

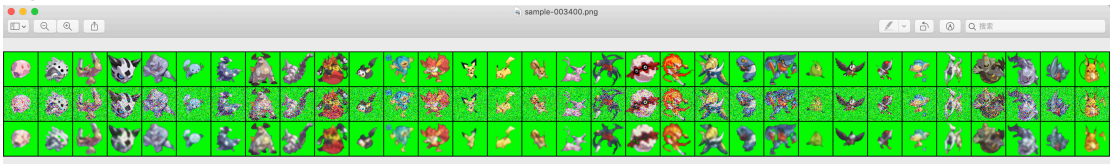




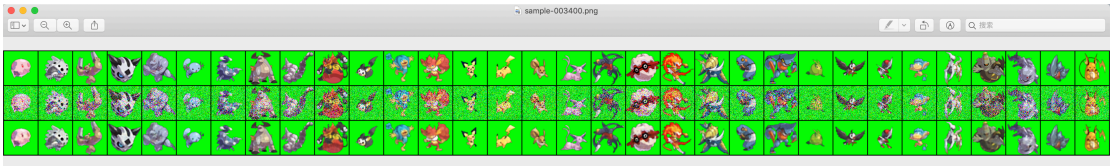


f) 1) I changed the activation function from ReLU to Sigmoid, the results become bad. The following images show the result.

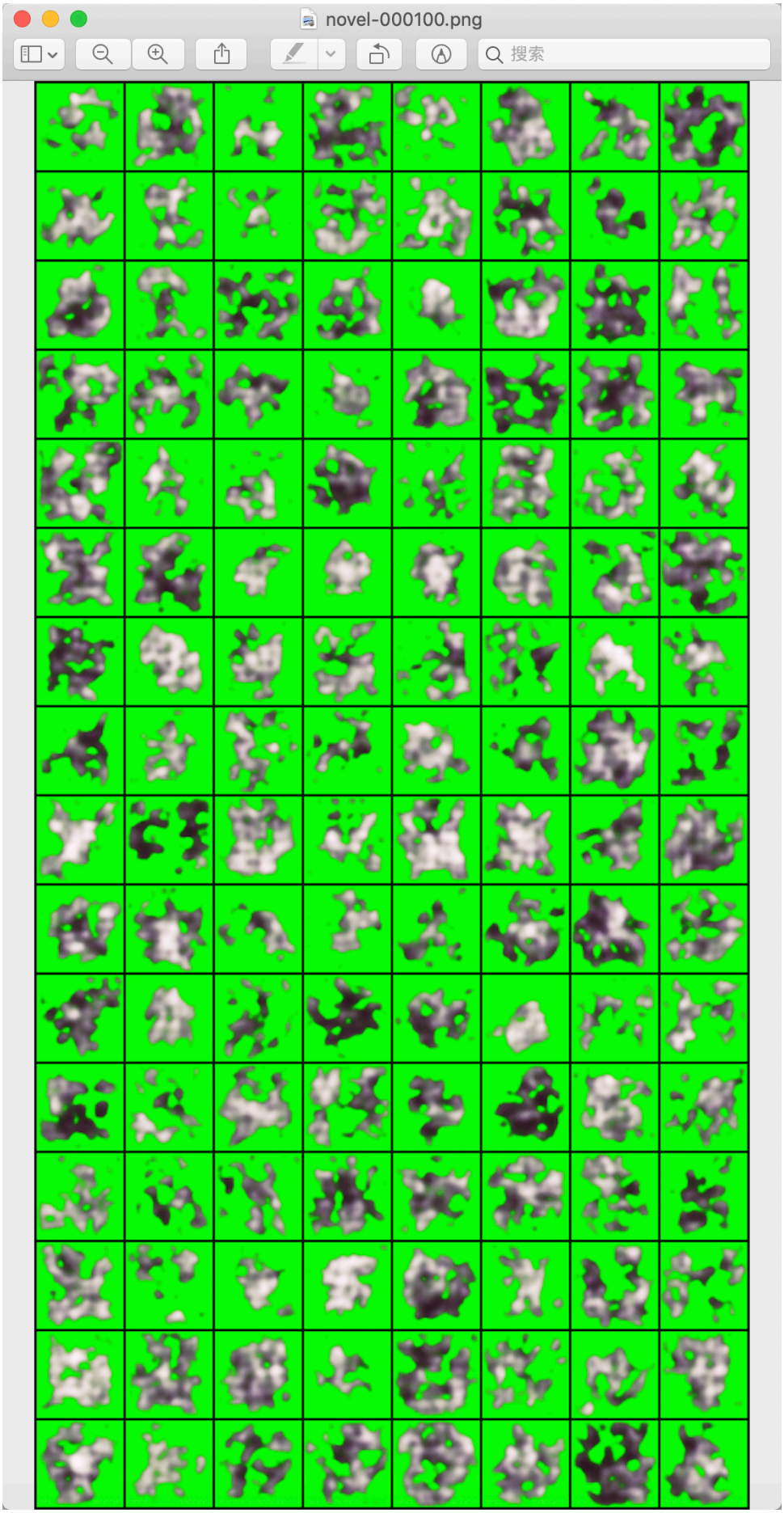
Sigmoid function



ReLu



2)The size of results are changed through the change of batch sizes(32 -> 128).


novel-000100.png
搜索