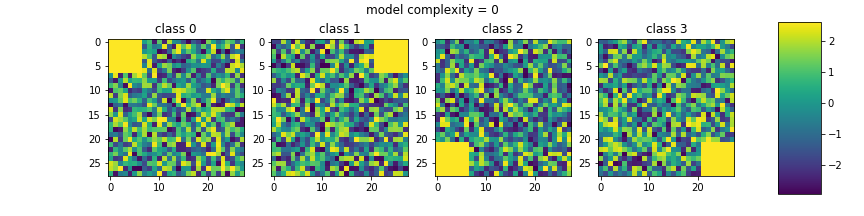
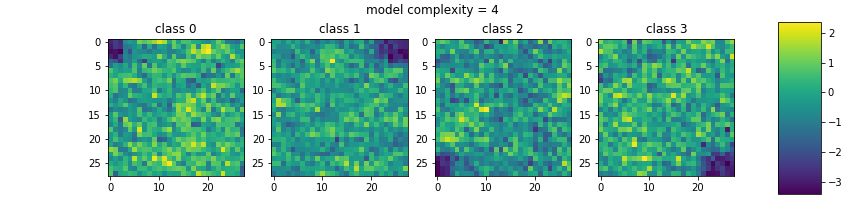
### PART I Data generation

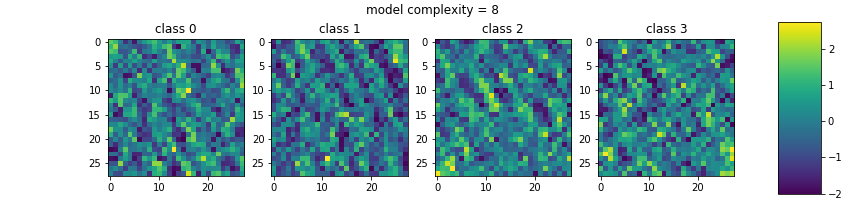
Demo images saved in folder:



Demo images:







**Total number of images**: 4000(1000 for each class)

**Image size**: 28x28

**Fixed region size**: 7x7

**Fixed region value**: 1

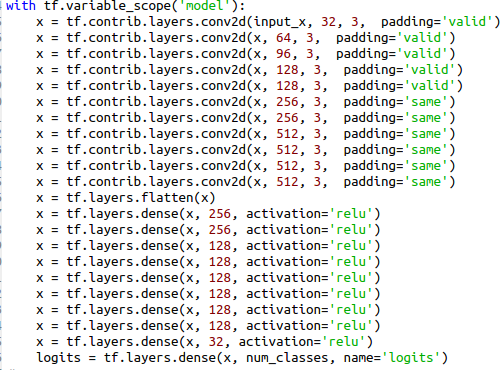
**Model complexity**: from 0-9, each number stands for the number of constitutional layers used.

**Notes**:

1. 为了保证使用更多卷积层生成的图像一定有更高的复杂度，需要逐渐增加卷积层的数量，按顺序生成不同复杂度的数据：每次使用一定数量的卷积层生成数据时都需要把所有weights冻结，然后在下一次使用更多卷积层生成数据时，恢复低层冻结的weights并随机初始化新加层的weights。
2. 只有在生成的数据的scale接近的情况下比较perturbation的绝对值才有意义。因此为了保证不同复杂度的数据有相似的scale，需要对所有生成的数据进行normalization。

### PART II model training

最复杂的模型为20层，其中前11层为conv，之后为全连接，所有层激活函数为relu。层数越多的模型使用的学习率越低，所有训练好的模型识别正确率都为99%以上。



### PART III comparison of data complexity

**Parameters**:

Images used to train the universal perturbation: 400(100 for each class)

Max iteration numbers to train the perturbation: 10

Fooling rate: > 70%

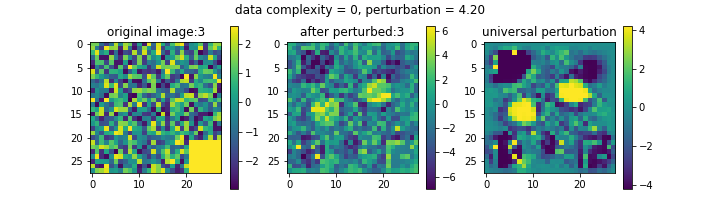
ξthe maximum magnitude of the perturbation vector

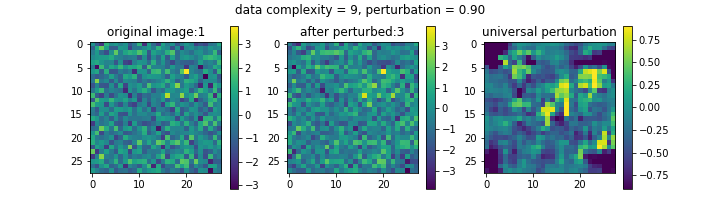
Method used to generate perturbation: Deep Fool

**Image Folders**:



**Image Demo**:





**Experiment Result**:

当perturbationξ小于表格中的数值时fooling rate在**Max iteration numbers**内无法超过70%

|  |  |  |
| --- | --- | --- |
| Data complexity | Model layers | ξ |
| 0(means use original images) | 4 | 4.2 |
| 1 | 2.0 |
| 2 | 1.8 |
| 3 | 1.8 |
| 4 | 1.8 |
| 5 | 1.6 |
| 6 | 1.6 |
| 7 | 1.6 |
| 8 | 1.1 |
| 9 | 0.9 |

数据复杂度在0-9之间时，准确率为99%以上，

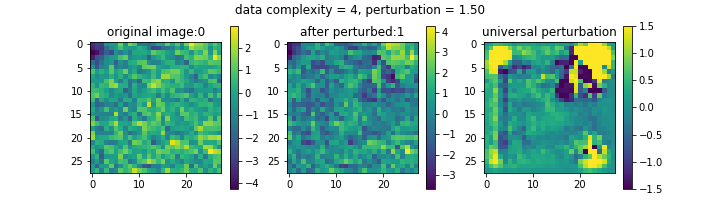
数据复杂度大于9之后模型准确率大幅下降（小于80%）

### PART IV comparison of model complexity

**Image Folders**:



**Image Demo**:



**Image命名规则**：

model-c4-l9-acc1.000000.pb-per1.5.png

C4表示data complexity=4

l9表示model layers=9

Acc1.0000000表示原始模型准确率100%

Per1.5表示最小perturbation=1.5

**Experiment Result**:

当perturbationξ小于表格中的数值时fooling rate在**Max iteration numbers**内无法超过70%

|  |  |  |
| --- | --- | --- |
| Model layers | Data Complexity | ξ |
| 1 | 4 | 2.7 |
| 3 | 2.4 |
| 6 | 1.5 |
| 9 | 1.5 |
| 12 | 1.3 |
| 15 | 1.2 |
| 18 | 1.5 |