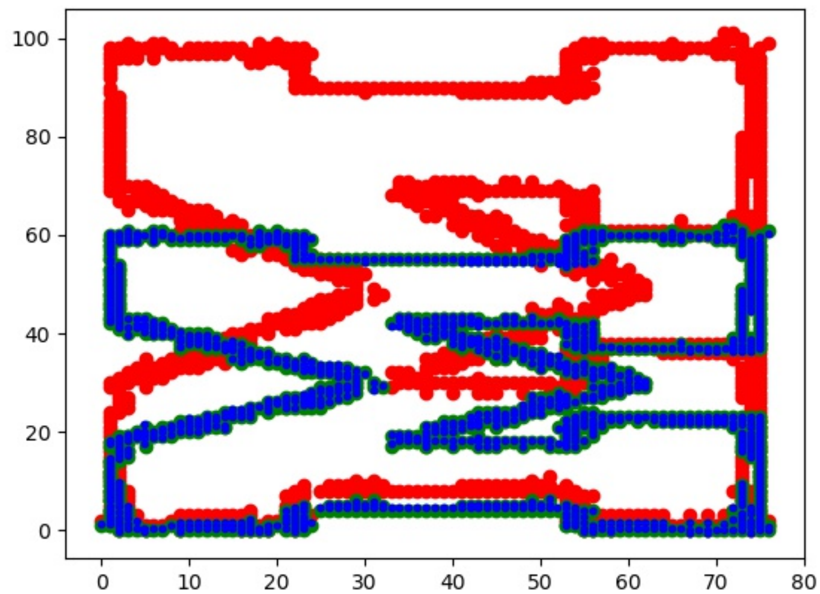
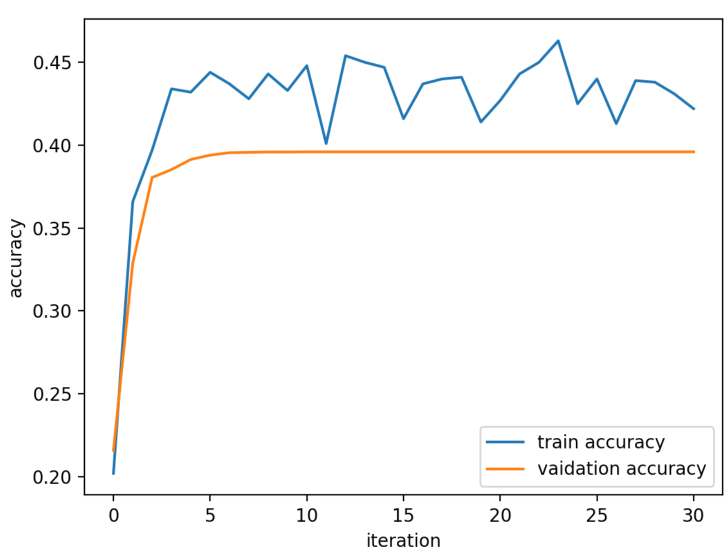
1. **Optimization and Fitting**
    1. implemented in the code
    2. learning_rate = 5e-5, epoch number =10000, batch_size =1, single layer net work with 2 node



2. Softmax Classifier with One Layer Neural Network

network architecture: one Fully connected layer with softmax loss
INPUT - FC - Softmax -OUTPUT
Dimension of Weight in FC : 3072x10 (num_classes)
Dimension of bias in FC : 10x1



learning_rate = 3e-2
lr_decay=0.3,
num_epochs=40,
batch_size=128

Test Accuracy : 39.5%

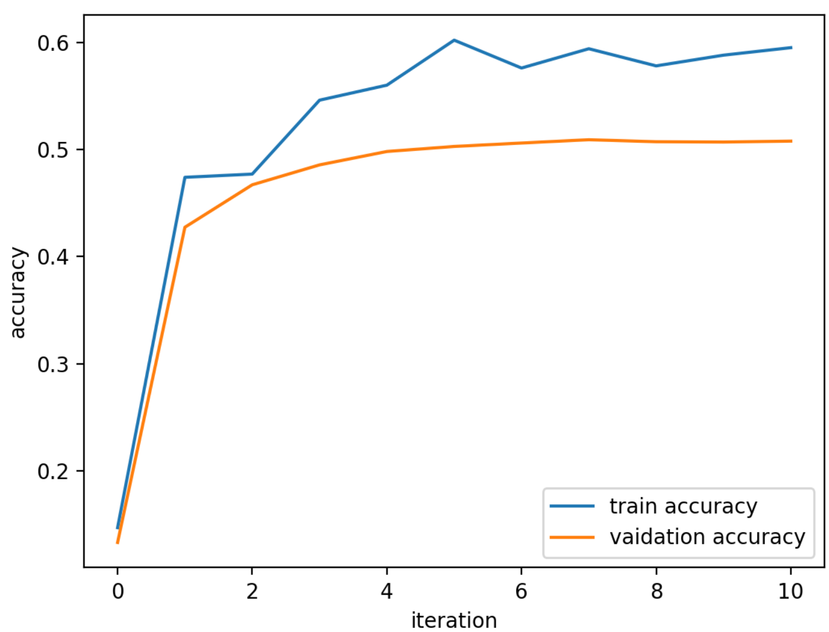
The below table shows the

parameter I chose for experimentation, and including my results and analysis. The test accuracies are shows across different range of learning rate and learning decay rate. As you can see the there is a sweet spot where the test accuracy is maximized across different rate and decay (local/global minima of test accuracy). To know the global minimum, more extensive parameter sweep including different batch size or number of epochs is required.

Epoch number = 30

learning rate\decay	0.3	0.4	0.5
2E-02	0.391	0.377	0.390
3E-02	**0.395**	0.383	0.387
4E-02	0.3844	0.381	0.385
5E-02	0.382	0.382	0.374

3. **Softmax Classifier with Hidden Layer**



the number of hidden dimension: 200
learning_rate =
1e-1 ,#5e-3,
lr_decay=0.7,
num_epochs=10,
batch_size=128

network architecture : fc - relu - fc - softmax
W1: 3072x200
b1 : 200x1
W2: 200x10
b2: 10x1

test accuracy: 51.8%

Similar to number 2, the test accuracies were tested across different learning rate and decay. Across different ranges of hyper parameters, there was a sweet pot where the test accuracy was maximum. Some of the fixed learning rate, as learning rate was increased the test accuracy was increased. But this trend was different on different range of rate (2E-01). To know the global minimum, more extensive parameter sweep including different batch size or number of epochs is required.

Epoch number = 10

learning rate\decay	0.5	0.6	0.7
5E-02	0.467	0.485	0.4966
1E-01	0.503	0.511	**0.518**
2E-01	0.517	0.516	0.514
3E-01	0.513	0.515	0.510

4. Fooling Image

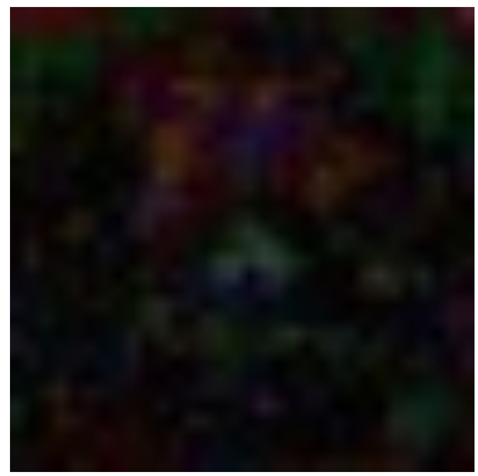
1. code implemented

2. It originally classified as airplane (0), which is a correct label for this test image. After the image is modified (fooling) the image was classified as a target fooling class which is deer (5). The below figures show the original image, and the image which is modified after the image was changed from gradient ascent. After it is fooled, the greenish pixels was added to the background (which make sense, where many deer images has grass in the background). Also the difference was also depicted, where it was scaled by 100, since the change was too small to visualize.

original

fooling

difference



Robustness of this network : In order to fool the image the there were some color change as shown in the fooling image (around the edge), so the **network is weak to color change**. However it is unclear if it is robust to shape change since the shape of the object almost remains the same.