CSC-498: Convolutional Neural Networks
Final Project #2: Fooling Convolutional Neural Nets
Akshay Kashyap

This project aims to show how a convolutional neural net can be fooled by generating images using a genetic algorithm. This is an implementation of the paper "Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images" by Nguyen A., Yosinski J., Clune J.

The code population starts off with a Convolutional Neural Net pre-trained on MNIST (implemented using Keras) and then uses a Genetic Algorithm to generate images that closest get predicted to certain target MNIST images.

1. **Fitness**: We use Softmax function for fitness:

$$L_i = -f_{y_i} + \log \sum_j e^{f_j}$$

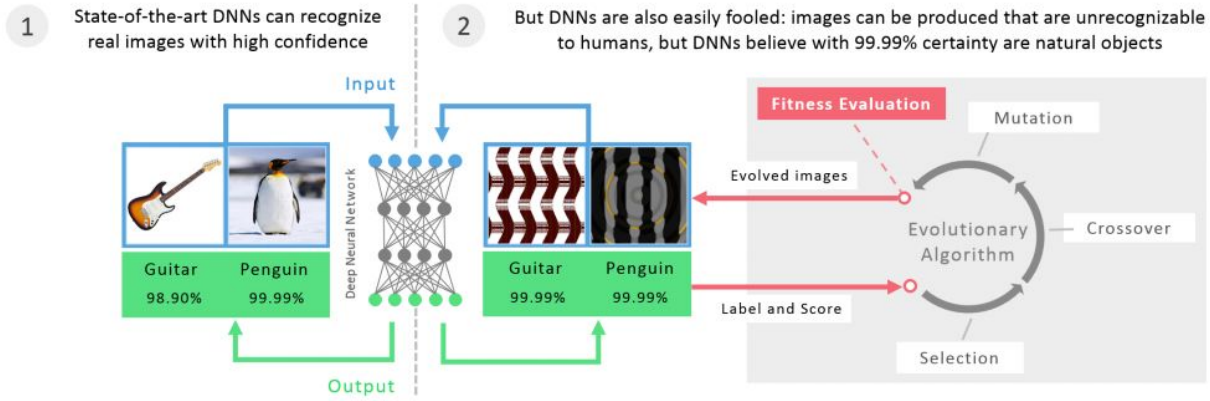
Where y_i is the target prediction (the class we want the neural network to falsely predict) and j is all other classes. f_{yi} and f_j are scores for those classes. The softmax function measures loss between the scores predicted for each of class for the genetically evolved image and the target class we want it to predict for that image. The score would be inverse loss:

$$S = \frac{1}{Loss}$$

This way, we can measure how close (or far) our evolved image is getting to the target, and thus evolve it accordingly.

2. **Mutation**: Choose a random subsets of pixels and replace them with random pixels between 0 - 255.

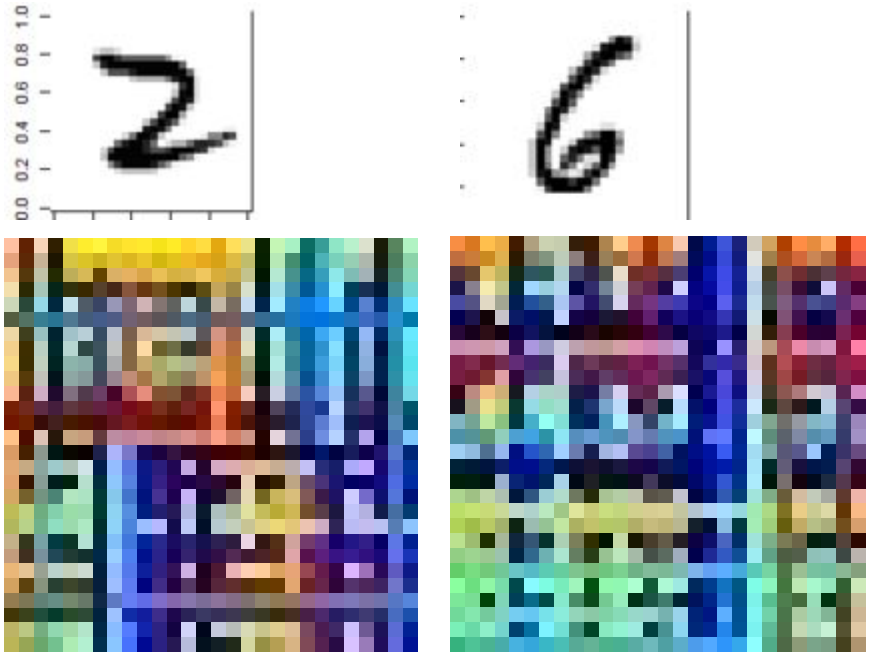
The process is described by the image below,



For a particular target MNIST class (from 0 to 9) we start off with a random population, evaluate fitness by getting a prediction for images in that epoch and calculating score as shown above, and evolve population accordingly. Repeat this for a set no. of epochs.

Results:

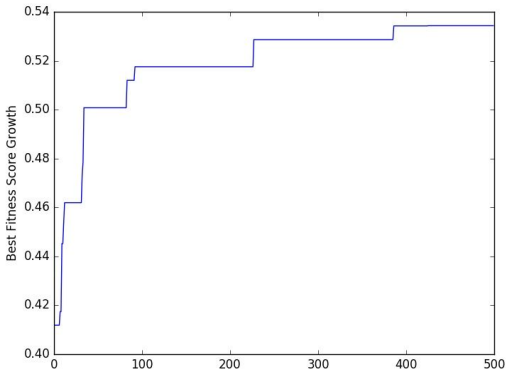
Some of the highest-scoring evolved images:

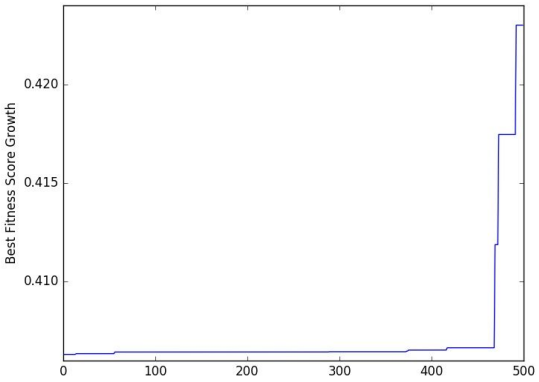


Class: **2** Class: **6**

We can see that the evolved images do vaguely have features resembling that of actual MNIST images for these classes. The first one has some features similar to 2, and the second one looks like an inverted 6.

The images below show the growth in fitness scores for these:






```
Successfully Fooled  4 /10 times

Actual | Predicted | Score
(0, 0, 0.53430795743715709)
(1, 2, 0.40668167423922313)
(2, 2, 0.68439235128680576)
(3, 2, 0.40659182502873892)
(4, 2, 0.40662528014050564)
(5, 0, 0.40657945710891213)
(6, 6, 0.42299878895346543)
(7, 7, 0.49387428515579856)
(8, 2, 0.48565623348280618)
(9, 0, 0.40661690416565605)
```

Conclusion:

In conclusion, this raises some questions about true generalization by these CNNs. A theory discussed in the paper is how Generative Models could possibly be better than Discriminative models. CNNs (and DNNs in general) are usually Discriminative, in that they calculate p(y | X), where y is the class and X is the data point, but Generative Models calculate P(y, X). Generative Models may be harder to fool since they would also know P(X) before calculating P(y | X), and would understand that these evolved images have a very low possibility of occurring in any given task/training.