Applied Evolutionary Algorithms
Evolutionary design of adversarial examples for neural networks

Ladislav Ondris (xondri07)

May 4, 2022

1 Description

The aim of this work is to experiment with different parameters of a genetic algorithm (GA) for finding adversarial examples that cause a convolutional neural network (CNN) misclassify an object in an image.

The adversarial examples are created by adding random values to pixels such that it is still visible what the image contains. A set of random values is called perturbation. This work searches for targeted perturbation, which means its goal is to increase the probability of a specific class rather than just making it misclassify the object.

An easy dataset was selected for this task, which enables faster experimentation. The work can, however, be easily extended to arbitrary classification dataset. The dataset is called Fashion MNIST. It consists of 60k training samples and 10k testing samples divided into ten classes. All images are grayscale with resolution of 28x28 pixels.

While searching for perturbation, the method considers only a smaller portion of the dataset, e.g. 100 samples, with roughly balanced classes. This means that the perturbation should work universally for any given image. This is a harder task as opposed to finding image specific perturbations.

The sought perturbation should not only increase the confidence of a specific class, but also be as little conspicuous as possible. Thus, the fitness function consists of two parts. A special measure was proposed by the authors of POBA-GA [1] that corresponds to how humans perceive the perturbations.

2 Target Model

A simple CNN was trained with 90% accuracy on the Fashion MNIST classification dataset and has 622k trainable parameters. The model is used as a black box because the method does not require any information about its internal structure.

3 Languages, Tools & Libraries

All code is written in Python. The model was trained in the TensorFlow library. Matplotlib is used for creating plots displaying experiment results. The code running experiments uses an

implementation of the genetic algorithm from the PyGad library[1].

4 Experiments & Results

The genetic algorithm can be used to successfully find adversarial examples, as Figure 1 demonstrates.

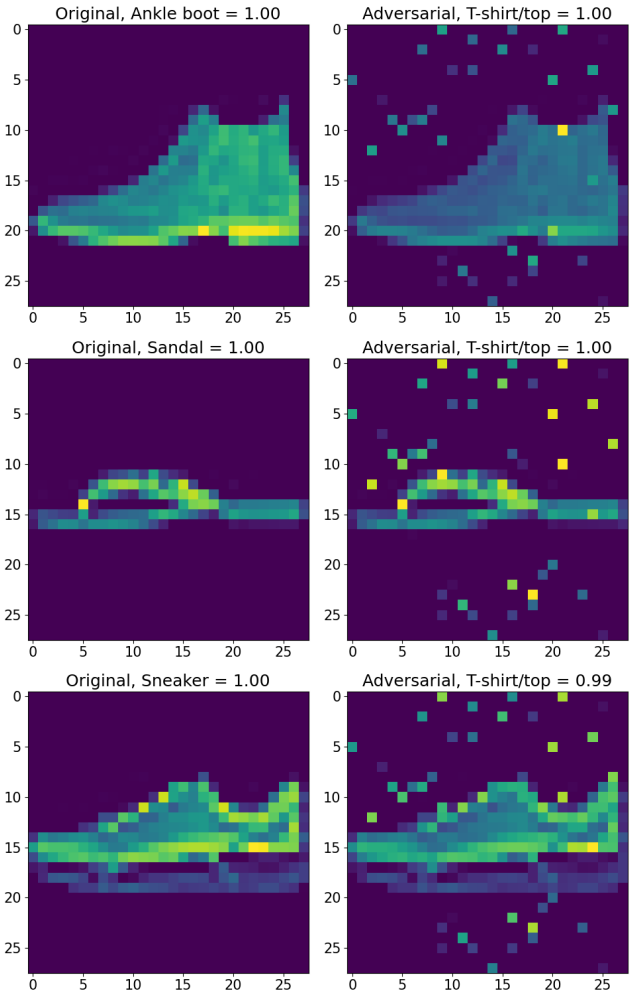

Figure 1: The same perturbation was added to different images, all of which resulted in being predicted as the same class.

100 images were used throughout all experiments for finding perturbations with highest fitness, while another 100 images were used for final accuracy evaluation.

A number of experiments were conducted to determine the best parameters of the genetic algorithm—population size, mutation probability, number of genes mutated, crossover probability. The process was repeated five times for each parameter setting for consistent results.

An experiment for determining the appropriate number of generations was executed, as depicted in Figure 3, which shows that even after 60 generations, the fitness is still slowly increasing.

[1] https://pygad.readthedocs.io/en/latest/

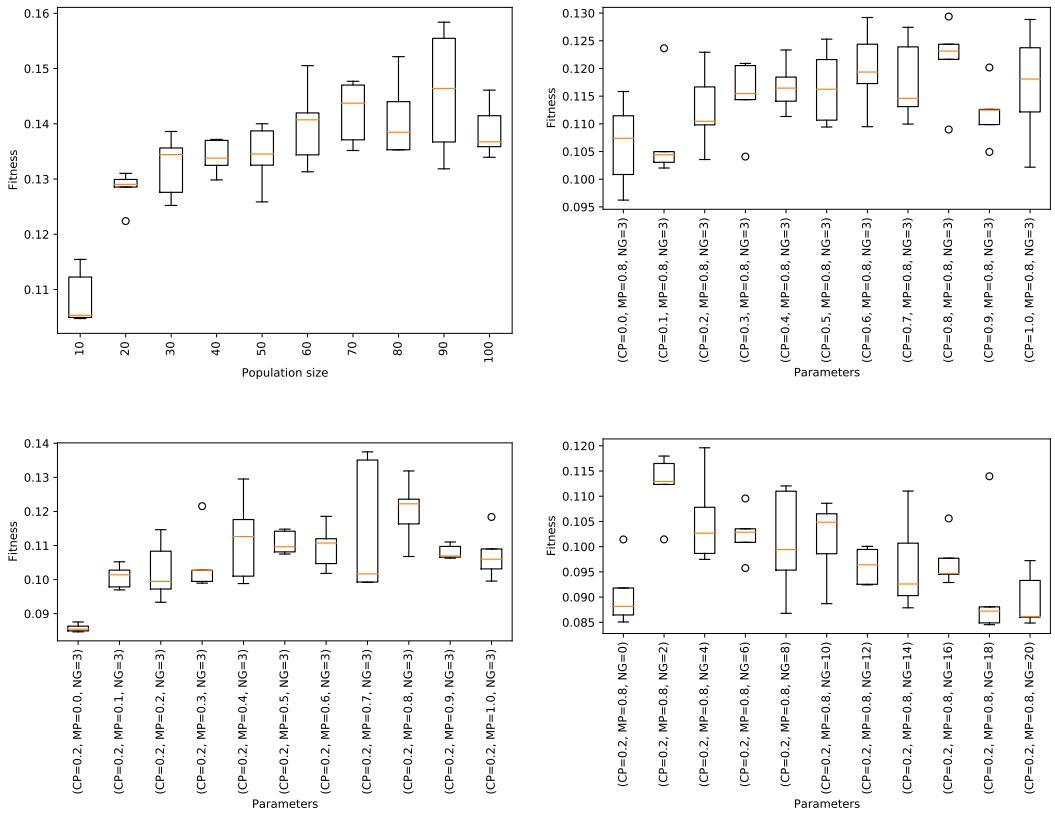

Figure 2: The figure shows the relationship between parameter values and the fitness.

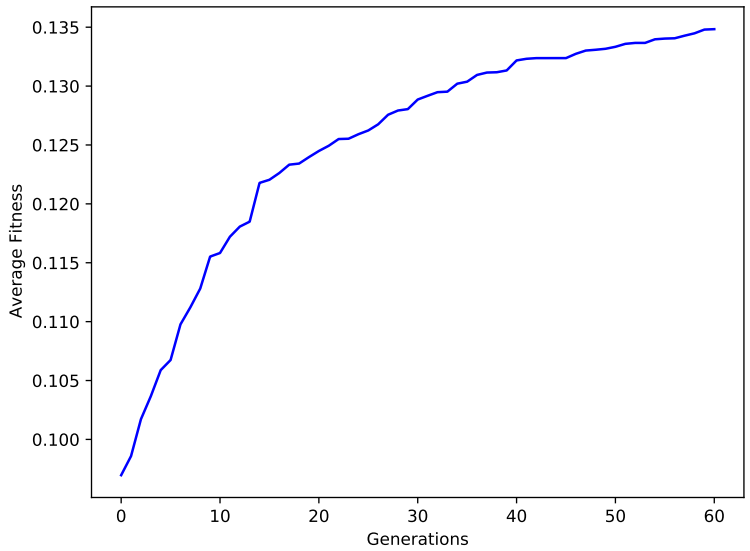

Figure 3: The figure shows the relationship between the number of generations and fitness value.

Even if the fitness is still increasing, the perturbation example may already be sufficient. The probability of successful attack is certainly an important factor.

References

[1] J. Chen, M. Su, S. Shen, H. Xiong, and H. Zheng. POBA-GA: perturbation optimized black-box adversarial attacks via genetic algorithm. *CoRR*, abs/1906.03181, 2019. URL `http://arxiv.org/abs/1906.03181`.