Computational Intelligence for Optimization

MASTER'S DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS – MAJOR IN DATA SCIENCE

Breaking CNNs – MNIST Dataset

Oreo Group

Geraldo Timbe, M20200603

Manuel Carreiras, M20200500

Venâncio Munhangane, M20200579

May, 2021

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

TABLE OF CONTENTS

TABLE OF FIGURES

1. Introduction

On deep neural networks the convolutional neural network (CNN) has become the state-of-the-art method for image recognition [1]. However, this method is sensitive to small perturbations, meaning that they can be easily fooled by adversarial attacks and classify images incorrectly.

Genetic Algorithm (GA) is one of the algorithms able to fool CNNs from classifying images correctly by generating adversarial examples. GA is an optimization technique inspired by the evolutionary process in biological cells and is based on Darwin's theory where the selected population is the survival of the fittest [2].

GA is a robust optimization mechanically quite simple, involving nothing more than random number generation, string copies and partial string exchanges. This robust optimization uses three main operators namely selection, crossover and mutation.

On this report it is going to be shown the effect of GA by creating adversarial examples. These images are going to be created using MNIST dataset. The test dataset is going to be evolved using the main operators of GA and the goal is to make CNN model classify images incorrectly. This paper is organized as follows. Section 2 shows the GA Operators, the fitness function and the constants used in this project. Section 3 describes the results of our experiments. On Section 4 and 5 are presented the conclusions and the propose for future work respectively.

2. Used operators and fitness function

2.1. GA Operators

In this project was implemented the three main operators (selection, crossover, and mutation) of the GA and also elitism. The selection process was performed in three different ways namely: fitness proportion selection, tournament selection and ranking selection. The crossover was implemented in two different ways namely: single point crossover, and arithmetic crossover. And lastly for mutation was used inversion mutation and swap mutation.

The adversarial images were created by combining the operators described above. The combination of the operators resulted on the 12 scenarios and for each one was analysed their classifications results using one well trained CNN model, to see the effects of the GA operators. After analysed all the results of the 12 scenarios was selected the one that make the CNN model perform worst and was applied the elitism to see the effect.

2.2. Fitness function

The key step to perform everything described above is defining the fitness function to calculate the fitness which is the driven force for GA and depends on the nature of the problem. In this project the fitness function was defining as follows:

$$FitnesValue = (\text{abs}(-(0.0000005 * \text{linalg.norm}(\text{image} - \text{target\_image}) + (\text{target\_score} - \text{score})))$$

Where:
image: is the predict label by the model;
target_image: is the original label of the image;
target_score: Is the desirable probability that the model want to predict in this case 1;
score: Is the maximum probability given by the model for the predict label;
linalg.norma: It is equivalent to Euclidean distance;
0.0000005: is a constant used in fitness algorithm, multiplies the distance between 2 images.

Basically, the function above measures the distances between two images and when the value is closer to zero means that the images are similar, in this case the original image and the adversarial images are similar. Because the goal on this project is to create adversarial images that are different from the original and can misclassify the model this is a maximization problem because where the desirable is to maximize the distance between the original and the adversarial images.

One off the short comings of the GA is that they are slow, as nature evolution, cannot happen fast. The test dataset of MNIST dataset contains 10000 images. Using a CPU would take a while to evolve all, in through 100 generation. These factors led to a computational limitation and for that reason, GA was implemented using the constants bellow:
N=1000: On the test dataset, 1000 were selected randomly to be evolved;
G=10: Number of generations;
Pob_cross=0.6: Probability of implement crossover;
Prob_mut=0.5: Probability of mutate the offspring;
Tournement_size = 20: Number of individuals to be randomly selected on the tournament selection.

3. Results

After defining the fitness function and set the constants of the operators, the first approach was train the model to be break. As said before this well-trained model was used to classify the adversarial images resulted from each combination of the operators. The performance of the model on each of the 12 scenarios was analysed using the classification report and plotting the accuracy, precision, recall and f1-score on each generation. This approach allowed to compare and understand the effect of the GA operators.

3.1. Trained model

The figure 1 provides the performance of the trained model in each label. In general, the model is well trained since it is able to predict correctly all the labels (0-9). The accuracy of the precision, recall and f1 score are 0.95%. On the following results it is going to be analyzed the performance of this model on the adversarial images.

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.99	0.97	0.98	1135
2	0.98	0.93	0.95	1032
3	0.93	0.95	0.94	1010
4	0.95	0.96	0.96	982
5	0.92	0.94	0.93	892
6	0.98	0.93	0.96	958
7	0.97	0.96	0.96	1028
8	0.87	0.95	0.91	974
9	0.95	0.92	0.93	1009
micro avg	0.95	0.95	0.95	10000
macro avg	0.95	0.95	0.95	10000
weighted avg	0.95	0.95	0.95	10000
samples avg	0.95	0.95	0.95	10000

Figure 1: Performance of the trained model on test dataset.

3.2. First scenario: Fitness proportion selection, single point crossover and swap mutation.

Was created adversarial images resulted from the combination of the following GA operator's: fitness proportion selection, single point crossover and swap mutation (FPS_SPC_SM). The figure 2 shows the classification report which has the results of the performance of the well-trained model after evolving the pictures through 10 generations and the plot on the right shows the accuracy, f1_score, recall and precision. In general, it is possible to see that before evolving the GA the model is classifying correctly the images but as long you start evolving through 10 generations the model start misclassifying the images for all the labels and end up with 0.15 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.31	0.03	0.05	192
1	0.00	0.00	0.00	0
2	0.40	0.07	0.12	283
3	0.03	0.50	0.06	2
4	0.80	0.08	0.15	50
5	0.00	0.00	0.00	3
6	0.29	0.03	0.06	152
7	0.36	0.02	0.04	181
8	0.10	0.90	0.19	92
9	0.38	0.56	0.45	45
micro avg	0.15	0.15	0.15	1000
macro avg	0.27	0.22	0.11	1000
weighted avg	0.35	0.15	0.10	1000
samples avg	0.15	0.15	0.15	1000

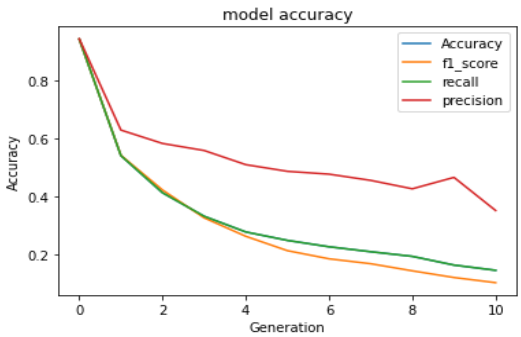

Figure 2: Performance of the trained model on adversarial images using FPS_SPC_SM.

3.3. Second scenario: Fitness proportion selection, single point crossover and inversion mutation.

Was created adversarial images resulted from the combination of the following GA operator's fitness proportion selection, single point crossover and inversion mutation (FPS_SPC_IM). In general, it is possible to see the same behavior described on the first scenario. Before the evolving on GA, the model is correctly classifying most of the images but as long start evolving through 10th generation the model start misclassifying the images for all the labels and ends up with micro average score of 0.20 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.27	0.04	0.07	80
1	0.00	0.00	0.00	0
2	0.28	0.11	0.16	201
3	0.00	0.00	0.00	32
4	0.07	0.01	0.02	80
5	0.10	0.12	0.11	25
6	0.28	0.09	0.13	114
7	0.20	0.02	0.04	134
8	0.22	0.75	0.34	195
9	0.12	0.06	0.08	139
micro avg	0.20	0.20	0.20	1000
macro avg	0.15	0.12	0.09	1000
weighted avg	0.20	0.20	0.14	1000
samples avg	0.20	0.20	0.20	1000

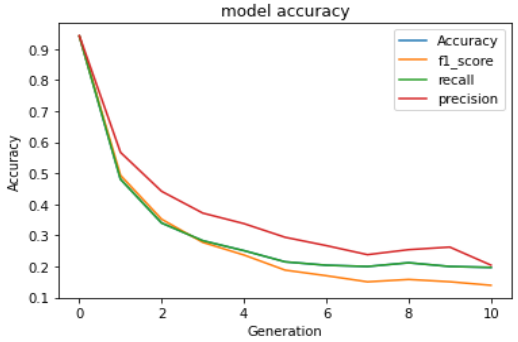

Figure 3: Performance of the trained model on adversarial images using FPS_SPC_IM.

3.4. Third scenario: Fitness proportion selection, arithmetic crossover, and swap mutation.

For this scenario was created adversarial images resulted from the combination of the following GA operator's fitness proportion selection, arithmetic crossover, and swap mutation (FPS_AC_SM). Before evolving on GA the model is classifying correctly most images but when the evolving process start the performance decrease for all the labels and ends up with micro average score of 0.15 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	30
1	0.00	0.00	0.00	0
2	0.32	0.03	0.06	347
3	0.10	0.67	0.17	3
4	0.00	0.00	0.00	57
5	0.11	0.06	0.08	16
6	0.64	0.07	0.13	193
7	0.91	0.05	0.09	201
8	0.12	0.84	0.21	114
9	0.23	0.49	0.31	39
micro avg	0.15	0.15	0.15	1000
macro avg	0.24	0.22	0.11	1000
weighted avg	0.44	0.15	0.10	1000
samples avg	0.15	0.15	0.15	1000

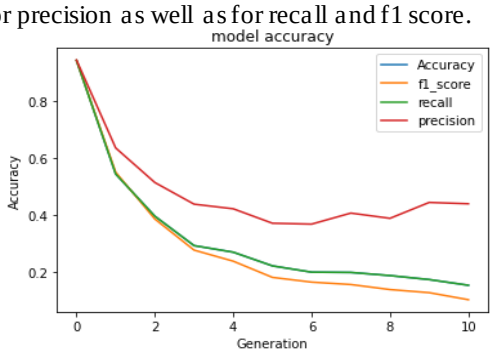

Figure 4: Performance of the trained model on adversarial images using FPS_AC_SM.

3.5. Fourth scenario: Fitness proportion selection, arithmetic crossover, and inversion mutation.

For this scenario was generated adversarial images using the combination of the following GA operator's fitness proportion selection, arithmetic crossover, and inversion mutation (FPS_AC_IM). In general, it is possible to see the same behavior described on the previous scenarios. After the 10th generation the model ends up with micro average score of 0.19 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.11	0.02	0.03	55
1	0.00	0.00	0.00	0
2	0.19	0.05	0.08	240
3	0.20	0.25	0.22	44
4	0.00	0.00	0.00	15
5	0.06	0.08	0.07	26
6	0.29	0.07	0.11	198
7	0.53	0.06	0.11	126
8	0.20	0.76	0.32	183
9	0.04	0.03	0.03	113
micro avg	0.19	0.19	0.19	1000
macro avg	0.16	0.13	0.10	1000
weighted avg	0.23	0.19	0.13	1000
samples avg	0.19	0.19	0.19	1000

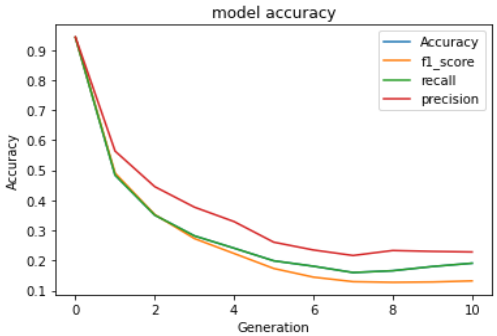

Figure 5: Performance of the trained model on adversarial images using FPS_AC_IM.

3.6. Fifth scenario: Tournament selection, single point crossover and swap mutation.

In this scenario the adversarial images resulted from the combination of the following GA operator's, tournament selection, single point crossover and swap mutation (TS_SPC_SM). In contrast of the results above it seems that the implementation of TS_SPC_SM until the10th generation does not create many adversarial images able to mislead the model. The performance of the model does not converge to zero at the same speed as on the on the previous scenarios. The model ended up with micro average score of 0.70 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	1.00	0.51	0.68	76
1	0.98	0.80	0.88	110
2	0.74	0.53	0.62	119
3	0.69	0.71	0.70	123
4	0.98	0.74	0.84	128
5	0.71	0.92	0.80	60
6	0.90	0.91	0.90	96
7	0.72	0.58	0.64	113
8	0.44	0.80	0.57	115
9	0.29	0.42	0.34	60
micro avg	0.70	0.70	0.70	1000
macro avg	0.74	0.69	0.70	1000
weighted avg	0.76	0.70	0.71	1000
samples avg	0.70	0.70	0.70	1000

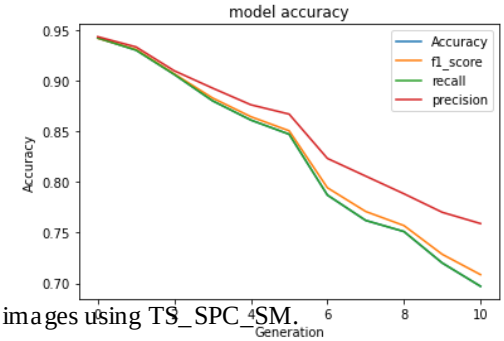

Figure adversarial images using TS_SPC_SM.

3.7. Sixth scenario: Tournament selection, single point crossover and inversion mutation.

In this scenario the adversarial images resulted from the combination of the following GA operators, tournament selection, single point crossover and inversion mutation (TS_SPC_IM). In general, it is possible to see different behavior described on the previous. The adversarial images resulted from operators TS_SPC_IM creates many adversarial images able to mislead the model. The performance of the model tends to converge to zero at rapidly and ended up with micro average score of 0.28 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.18	0.03	0.05	66
1	1.00	0.46	0.63	174
2	0.15	0.17	0.16	121
3	0.08	0.24	0.12	45
4	0.70	0.25	0.37	113
5	0.24	0.18	0.20	131
6	0.05	0.06	0.05	69
7	0.41	0.18	0.25	71
8	0.24	0.62	0.34	125
9	0.25	0.21	0.23	85
micro avg	0.28	0.28	0.28	1000
macro avg	0.33	0.24	0.24	1000
weighted avg	0.40	0.28	0.29	1000
samples avg	0.28	0.28	0.28	1000

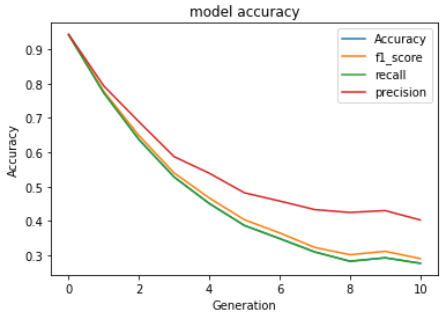

Figure 7: Performance of the trained model on adversarial images using TS_SPC_IM.

3.8. Seventh: Tournament selection, arithmetic crossover, and swap mutation.

In this scenario the adversarial images resulted from the combination of the following GA operators, tournament selection, arithmetic crossover, and swap mutation (TS_AC_SM). It is possible to see that the TS_SPC_IM does not create many adversarial images able to make the model misclassify. Throughout 10 generation the performance of the model tends to not converge to zero rapidly and ended up with micro average score of 0.65 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.96	0.67	0.79	79
1	0.97	0.89	0.93	157
2	0.76	0.44	0.56	103
3	0.65	0.63	0.64	136
4	0.95	0.53	0.68	110
5	0.53	0.67	0.59	49
6	0.74	0.74	0.74	35
7	0.79	0.53	0.63	152
8	0.27	0.75	0.40	75
9	0.50	0.69	0.58	104
micro avg	0.65	0.65	0.65	1000
macro avg	0.71	0.65	0.65	1000
weighted avg	0.74	0.65	0.67	1000
samples avg	0.65	0.65	0.65	1000

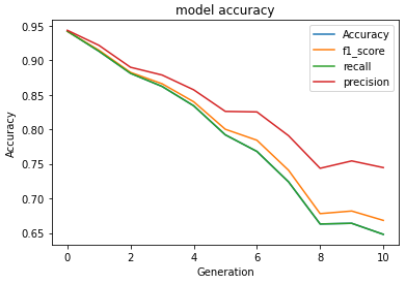

Figure 8: Performance of the trained model on adversarial images using TS_AC_SM.

3.9. Eighth scenario: Tournament selection, arithmetic crossover, and inversion mutation.

Was created adversarial images resulted from the combination of the following GA operator's: tournament selection, arithmetic crossover, and inversion mutation (TS_AC_IM). In general, it is possible to see a different behavior with the previous scenario. TS_AC_IM creates many adversarial images able to mislead the model. The performance of the model tends to converge to zero rapidly and ended up with micro average score of 0.32 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.57	0.14	0.23	83
1	0.94	0.36	0.52	141
2	0.21	0.20	0.20	115
3	0.23	0.38	0.29	77
4	0.53	0.22	0.31	95
5	0.34	0.49	0.40	105
6	0.31	0.29	0.30	107
7	0.78	0.32	0.45	113
8	0.19	0.54	0.28	96
9	0.15	0.18	0.16	68
micro avg	0.32	0.32	0.32	1000
macro avg	0.43	0.31	0.32	1000
weighted avg	0.46	0.32	0.33	1000
samples avg	0.32	0.32	0.32	1000

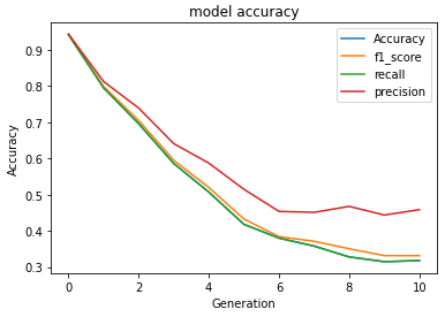

Figure 9: Performance of the trained model on adversarial images using TS_AC_IM.

3.10. Ninth scenario: Raking selection, single point crossover and swap mutation.

Was created adversarial images resulted from the combination of the following GA operator's, raking selection, single point crossover and swap mutation (RS_SPC_SM). In general, it is possible to see a different behavior with the previous scenario. RS_SPC_SM does not create many adversarial images able to mislead model. The performance of the model does not converge to zero rapidly and ended up with micro average score of 0.61 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.96	0.50	0.66	94
1	0.95	0.77	0.85	94
2	0.87	0.40	0.55	120
3	0.79	0.70	0.74	158
4	0.97	0.46	0.62	135
5	0.34	0.92	0.50	24
6	0.87	0.75	0.81	100
7	0.65	0.41	0.50	63
8	0.25	0.87	0.39	70
9	0.47	0.61	0.53	142
micro avg	0.61	0.61	0.61	1000
macro avg	0.71	0.64	0.61	1000
weighted avg	0.76	0.61	0.63	1000
samples avg	0.61	0.61	0.61	1000

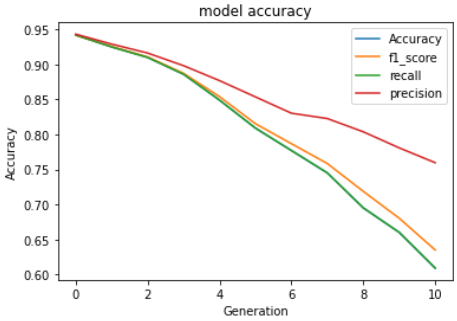

Figure 10: Performance of the trained model on adversarial images using RS_SPC_SM.

3.11. Tenth scenario: Raking selection, single point crossover and inversion mutation

The adversarial images resulted from the combination of the following GA operator's, raking selection, single point crossover and inversion mutation (RS_SPC_IM). In general, it is possible to see a different behavior with the previous scenario. RS_SPC_IM creates many adversarial images able to mislead the model. The performance of the model converges to zero at rapidly. At the end, the model ended up with micro average score of 0.26 precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	0.42	0.12	0.19	81
1	1.00	0.51	0.67	71
2	0.33	0.22	0.27	192
3	0.24	0.19	0.21	170
4	0.67	0.14	0.24	125
5	0.31	0.48	0.37	83
6	0.13	0.16	0.14	90
7	0.06	0.07	0.07	42
8	0.16	0.48	0.25	93
9	0.16	0.25	0.19	53
micro avg	0.26	0.26	0.26	1000
macro avg	0.35	0.26	0.26	1000
weighted avg	0.35	0.26	0.26	1000
samples avg	0.26	0.26	0.26	1000

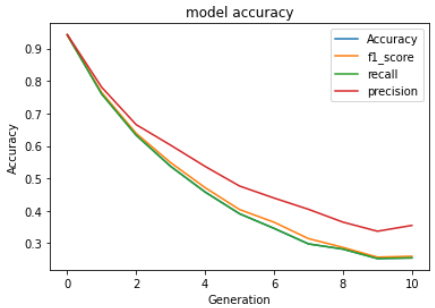

Figure 11: Performance of the trained model on adversarial images using RS_SPC_IM.

3.12. Eleventh scenario: Raking selection, arithmetic crossover, and swap mutation.

The adversarial images resulted from the combination of the following GA operator's, raking selection, arithmetic crossover, and swap mutation (RS_AC_SM). In general, it is possible to see a different behavior with the previous scenario. RS_AC_SM does not create many adversarial images able to mislead the model. Throughout 10 generations the performance of the model tends to not converge to zero rapidly, and ended up with micro average score of 0.62 for precision as well as for recall and f1 score.

	precision	recall	f1-score	support
0	1.00	0.83	0.91	47
1	0.98	0.66	0.79	98
2	0.94	0.51	0.66	223
3	0.52	0.57	0.54	131
4	0.88	0.40	0.55	121
5	0.51	0.84	0.64	45
6	0.86	0.73	0.79	131
7	0.84	0.65	0.73	79
8	0.30	0.84	0.44	91
9	0.30	0.62	0.40	34
micro avg	0.62	0.62	0.62	1000
macro avg	0.71	0.66	0.64	1000
weighted avg	0.77	0.62	0.65	1000
samples avg	0.62	0.62	0.62	1000

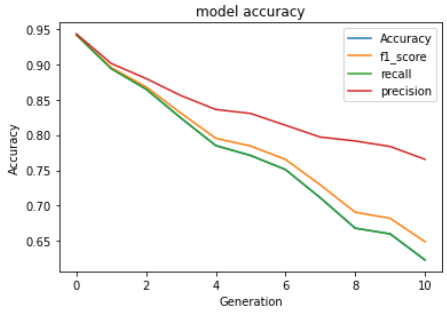

Figure 12: Performance of the trained model on adversarial images using RS_AC_SM.

3.13. Twelfth scenario: Raking selection, arithmetic crossover, and inversion mutation.

The adversarial images resulted from the combination of the following GA operator's, raking selection, arithmetic crossover, and inversion mutation (RS_AC_IM). it is possible to see a different behavior with the previous scenario. RS_AC_IM creates many adversarial images able to mislead the model. Throughout 10 generation the performance of the model tends to converge to zero rapidly, and ended up with micro average score of 0.27.

	precision	recall	f1-score	support
0	0.69	0.24	0.35	84
1	1.00	0.40	0.57	95
2	0.25	0.23	0.24	147
3	0.10	0.13	0.11	105
4	0.55	0.14	0.22	114
5	0.31	0.41	0.35	116
6	0.07	0.09	0.08	82
7	0.42	0.11	0.18	87
8	0.25	0.59	0.35	123
9	0.10	0.15	0.12	47
micro avg	0.27	0.27	0.27	1000
macro avg	0.37	0.25	0.26	1000
weighted avg	0.38	0.27	0.27	1000
samples avg	0.27	0.27	0.27	1000

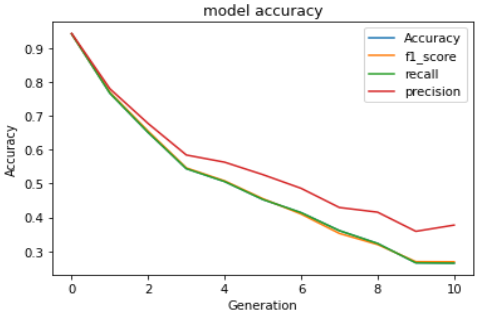

Figure 13: Performance of the trained model on adversarial images using RS_AC_IM.

3.14. Fitness on each scenario

After analysing the performance of the model on each scenario evolved by the GA's operators, was compared, the distribution of the fitness throughout 10 generations. The figure 14 contains thirteen box plots for each scenario.

Combining the results presented on each scenario with these box plots they clearly show that the scenarios with higher values of fitness are the ones with more adversarial images. These scenarios are able to misclassify the model. with this we can conclude that some operators succeed to maximize the fitness values throughout the 10 generations. On the other hand, operators that did not create many adversarial images to misclassify the model may need more generations to succeed.

To sum up the following combination fitness proportion selection, arithmetic crossover and swap mutation end up with highest fitness value and with many adversarial images able to misclassify the model . On the next section it is going to be applied the elitism on these operators using the same list of the individuals to see effect on the results.

3.15. Elitism on FPS_AC_SM

As said before the combination of the operators FPS_AC_SM (the green box-plot on figure 14) were considered the best for maximize the fitness. The maximum value of the fitness was verified on the last generation and was 653.74 (out [93] on the notebook) and the elitism was applied. As expected the elitism increased a bit the value o f the fitness to 656.31(out [93] on the notebook). Because we only maintained only 10 individuals it is normal no t having to much impact taking on consideration the size of the N (1000 individuals) (the black box-plot on figur

e 14). If the elitism were implemented for much generation the total fitness of the final population certainly woul d increase.

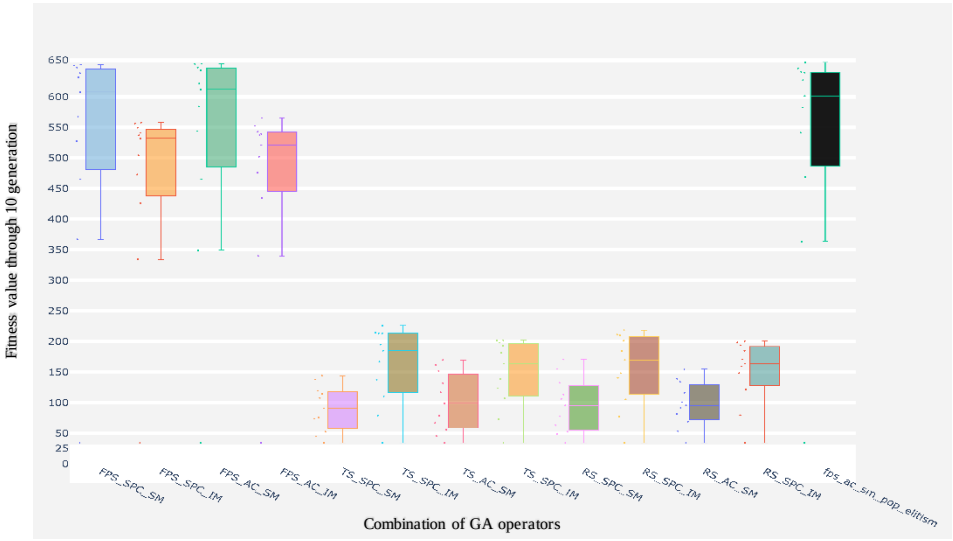

Figure 14: Distribution of the fitness on each of 12 scenarios through 10 generation.

4. Conclusions

After all the analysis it is clear that GA's operators are able to create adversarial images able to make a one well trained model misclassify. With the presented results and the figure 15 we can conclude that the implementation of the GA's operators on MNIST dataset was achieved and the main outcomes of this implementation are:

- In general, for the constants chosen on this project the evolving using FPS got many images able to misclassify the model;
- Some combinations of operators like TS_SPC_SM, RS_AC_SM need more generations or increasing the probability of crossover and mutation to create images able to misclassify the model;
- The elitism did not have much impact on the fitness due to the number of the generations meaning that the evolving process only kept 10 best individuals among 1000.

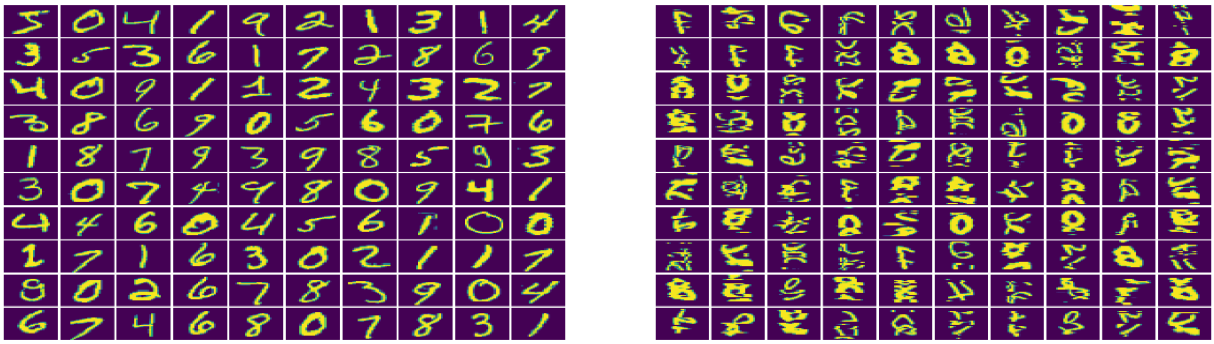

Figure 15: MNIST images Vs adversarial images from GA operators.

5. Future work

Due to computational limitation in this project was selected 10 as a Number of generations, 0.6 for probability of cross, and 0.5 for probability of mutation, so as future work the value of the probability may have to be reduced and increase the number of the generations and analyse the results.

6. Reference

[1] Vidnerová, P., & Neruda, R. (2016, August). Evolutionary generation of adversarial examples for deep and shallow machine learning models. In *Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016* (pp. 1-7).

[2] Hui, A., Huddin, A., Ibrahim, M., Hashim, F., & Samad, S. A. (2019, June). GA-Deep Neural Network Optimization for Image Classification. In *2019 International Journal of Advanced Trends in Computer Science and Engineering* (pp. 238-245). IEEE.