Adversarial Attack Via Multi-Objective Optimization

Jared Ratto
Department of Computer Science
University of Exeter, Exeter, UK

Internal Supervision:
Phoenix Williams
University of Exeter, Exeter, UK

***Abstract*—Adversarial examples have become a critical research topic for understanding the vulnerabilities of deep neural network classifiers to small perturbations. Many of these published attack methods have generated minimally perturbed images by optmizing or constraining a single l_p-norm. Methods that use the l_0-norm modify a sparse amount of pixels in an adversarial image while methods that use the l_∞-norm create a haze where all pixels are modified by a small amount. A novel attack method is proposed that optimizes both the l_0- and l_∞-norms as separate objectives. This algorithm returns a set of optimal adversarial examples for an input image with a varying number of pixels modified and with varying amounts of change added to each pixel. This project aims to offer a novel method for evaluating the robustness of deep neural network classifiers by visualizing vulnerabilities to l_0- and l_∞-norm perturbations via Pareto Fronts. A multi-objective attack method such as this one has the potential to reveal vulnerabilities of a deep neural network in ways that single-objective attack methods cannot. To evaluate the performance of the proposed method, images from the CIFAR10 dataset are studied on three neural network models, including two that have been trained on adversarially perturbed images.**

***Index Terms*—adversarial attack, multi-objective optimization, pareto front**

I. Introduction

Deep neural networks (DNNs) have achieved near-human like performance over the past decades when performing image recognition tasks. However, recent studies have demonstrated that small changes added to an image can cause the DNN to misclassify. These changes, known as perturbations, are often nearly imperceptible to the human eye and they do not affect the semantic value of the images. Existing works have even demonstrated that the modification of a single pixel can cause state-of-the-art DNN classifiers to misclassify [1]. Such images that are generated with the intention of misclassification are commonly referred to as adversarial examples. The vulnerability of neural networks to such perturbations poses serious security concerns, especially when the networks are employed in safety critical tasks [2], [3]. In the application of autonomous vehicles, an adversarial example that misclassifies a speed limit or traffic signal can lead to severe collisions and fatalities.

Since the initial work of C. Szegedy et al. [4], the generation of adversarial examples has become a critical area of research for evaluating the robustness of DNNs. The majority of existing methods formulate the generation of adversarial examples as an optimization task, where the process of solving the problem leads to the desired misclassification. Previously, much of the research explored what is known as white-box attacks (or gradient based attacks). In this setting, the attacker is assumed to have full access to the network and will use the model's gradients to find effective perturbations. Other works have later addressed the black-box scenario, where the attacker only has access to the outputs of the targeted DNN. The black-box scenario poses a challenge because the optimization problem becomes non-differentiable without the knowledge of the model's gradient. Optimization methods such as population based evolutionary algorithms or random search are often needed to find these optimal values [5], [6].

Most adversarial attack algorithms cause misclassification by adding a minimal amount of noise to an input image. To ensure the semantic content of the image is unchanged, works constrain the perturbation by its l_p-norm. Some commonly used norms include the l_0-norm which measures the number of pixels changed in an adversarial image, the l_2-norm which measures the average change to all of the pixels, and the l_∞-norm which measures the maximum change made to each pixel. Most attacks in the research seek to minimize just one of these norms without constraining the other norms. Therefore, an image perturbed by an l_0-constrained attack typically only has a few pixels changed but each is a changed by a large and noticeable amount. Likewise, an l_∞-constrained attack typically changes all pixels in an image but each by a nearly imperceptible amount. Meanwhile, l_2-constrained attack methods generate adversarial examples with clear distortions in some areas of the image [6].

Many published adversarial attacks generate a single, perturbed image by optimizing a single objective. However, some works use multiple objectives, such as multiple norms, to output a set of optimal solutions. Despite their multi-objective approach, these existing methods either aggregate objectives to converge to a single solution [7], or do not consider the trade-off between norms as an evaluation of a DNN's robustness [8]. To address these limitations, this project aims to improve the evaluation of DNN robustness by analyzing adversarial examples with varying norms. A set of adversarial examples with different trade-offs between norms would provide an improved understanding as compared to using a single solution, as the vast majority of existing methods use.

II. Related Works

A. Adversarial Examples

The security of DNNs has become a critical topic of research and has led to the study of many types of adversarial attacks. C. Szegedy et al. first observed the possibility of causing a DNN to misclassify by using a perturbation obtained from the model's training loss and gradient [4]. Another early white-box method, Fast Gradient Signed Model (FGSM), was developed and efficiently generated adversarial examples according to an l_∞-norm constraint [3], [9]. C&W and EAD were designed to use the logit layer in the DNNs rather than the loss function, while still having access to the model [10], [11].

Zeroth Order Optimization (ZOO) followed C&W's approach of using the model's logits but did so with only the outputs of the DNN in a black-box setting [12]. ZOO optimized adversarial examples by approximating the gradient though finite differences, but a weakness in this approach is that it required a very large number of queries. To address this, query-efficient attacks were developed such as GenAttack, which employed dimensionality reduction and evolutionary strategies in order to efficiently optimize without the model's gradient [3]. Square Attack used a random search algorithm for l_2- and l_∞-constrained attacks and achieved state-of-the-art performance in terms of efficiency and attack robustness, even at rates comparable to white-box attacks [6].

The previously listed works largely used an l_2 or l_∞ constraint and altered many pixels in the image; however, some works have focused on sparse attacks that minimize the l_0-norm and perturb only a few pixels. The most sparse of these is the One Pixel Attack, which found that it is possible to cause a DNN to misclassify by perturbing a single pixel. One Pixel found that this could occur for many small images, but the success rate of its algorithm greatly decreased when used on larger images from ImageNet [1]. Sparse-RS designed an l_0-constrained attack that achieved state-of-the-art success rates and query efficiency and that outperformed all blackbox and white-box attack methods [13].

B. Adversarial Robustness

Many methods of defense against adversarial attacks have been proposed and this remains an active area of research. Earlier methods relied on obfuscated gradients that yielded some protection in the whitebox setting. However, this type of defense can be overcome by stronger and gradient free attacks [6], [14]. Other types of defenses have included preprocessing techniques [15], detection algorithms [16], and adversarial training [17]. Among these, adversarial training has proven empirically to provide the most reliable defense against adversarial examples [18], [19]. As the name suggests, adversarial training involves training a classifier model with images that have been perturbed as adversarial examples. While this approach reliably increases robustness, one downside is that it is more computationally expensive to train such a network because it requires that an adversarial example first be generated. Various works have aimed to improve upon adversarial training to make training easier [19] and to further improve the resulting robustness [18], [20].

Strong attacks, such as those described in the previous subsection, are therefore encouraged as a means of effectively measuring adversarial robustness [19]. Stronger defeneses are compared against others based on their robust accuracy against strong attacks. In the research community, the website Robust Bench serves as a standard benchmark for ranking defended models [21]. Each model in the website's leaderboard is ranked based on its adversarial robustness to the AutoAttack algorithm. This paper will use some models included in this leaderboard during its experiment.

C. Multi-Objective Optimization

In order to introduce the multi-objective approach used in this project and used by others, we will briefly describe notable works on multi-objective optimization and their underlying concepts.

Multi-objective optimization is a practice that finds optimal solutions according to more than one objective. Generally, these objectives are conflicting, meaning that improving one objective often involves a trade-off with another objective. In principle, there exists no single solution that is optimal for all objectives; and thus, such methods return a set of optimal solutions that is commonly referred to as a Pareto Front or Pareto curve [22]–[24]. The concept of approximating and visualizing a Pareto Front for two objectives was introduced by S. Gass and T. Saaty [25], and by visualizing the curve one could analyze the trade-offs between objectives.

Seminal works on multi-objective algorithms include NSGA-II, which approximates the Pareto Front by using an evolutionary algorithm that sorts solutions into sets of non-dominated solutions. To ensure that the returned solutions are both optimal and diverse, a crowding operator is used to discourage solutions from being too similar to each other [22]. Another multi-objective evolutionary algorithm, MOEA/D, decomposes a multi-objective problem into a set of single objective problems to be optimized. The algorithm has robust performance particularly when the Pareto Front has a concave shape, which would make it otherwise difficult to approximate [26].

A few works on adversarial examples have used multi-objective optimization in their algorithms. The first of these was [8] and MOEA/D was used to calculate a Pareto Front of adversarial examples that minimized the l_0- and l_1-norms. A limitation of this work was that the shape and the trade-off relationship between the two norms were not deeply explored. Williams et al. used NSGA-II to simultaneously optimize solutions to misclassify and minimize the l_2-norm. However, the output of this work's algorithm converged to a single solution because of the domination strategy used [7].

III. Methodology

The algorithm in this work follows the general framework of NSGA-II, which uses an evolutionary algorithm. Evolutionary algorithms are optimization models that are inspired by natural selection in biology. Potential solutions to the optimization problem are represented as members of a population, and the effectiveness of a population member according to the problem's objectives is measured in a fitness function. Population members with a greater fitness have a greater probability of reproducing offspring that become the population members of the next iteration, referred to as a generation. During each generation, the offspring are created using an operator known as crossover, in which the offspring typically receive a random combination of the parents' attributes. To further increase the diversity of the population, offspring will undergo mutation, where some of its attributes are randomly changed according to a small, user-defined probability. The less fit members of the population are then typically replaced by those that are more fit [3].

In the conclusion section of [7], Williams et al. suggest that a useful opportunity for future research would be modifying their algorithm so that it minimizes the l_∞-norm, and that it minimizes each norm as independent objective rather than setting one of them as a constraint. By formulating it as a bi-objective problem for minimizing loss and the l_2-norm, their algorithm can be usefully changed to a multi-objective representation. The algorithm in our work therefore has followed a similar framework to theirs, but it has been significantly altered to make it multi-objective and for it to consider the l_∞-norm, while still following the general framework of NSGA-II. The objective function of this work is to independently minimize the l_0- and the l_∞-norms while requiring a negative loss as a constraint. In other words, each perturbed image returned by the algorithm must be an adversarial example. The l_0- and the l_∞-norms were chosen because they are generally conflicting objectives when creating adversarial images. Greatly reducing one of these norms generally comes at the expense of increasing the other in order for the image to still cause misclassification.

More formally, the objective function can be defined as follows:

$$\begin{aligned} &\text{minimize} \quad \mathrm{F}(x_{adv}) = (||x_{adv} - x||_0, ||x_{adv} - x||_\infty)^T \\ &\text{subject to} \quad max_{j \neq t}\{f_j(x_{adv})\} - f_t(x_{adv}) \leq 0 \end{aligned}$$

Where x is the original image inputted to a classifier, x_{adv} is the perturbed image, $f_j()$ is the probability of the classifier predicting the correct class of x, and $f_t()$ is the probability of the classifier predicting any other class besides the correct class. Collectively, $max_{j \neq t}\{f_j(x_{adv})\} - f_t(x_{adv})$ refers to the loss function of the classifier model with regards to the solution.

The constraint that the probability of the correct class be less than the probability any of any other class is enforced in the algorithm's domination criteria. If a solution is found to be adversarial from the loss function, it will automatically dominate any other solution that is not adversarial. If two solutions are both found to be adversarial, then the domination procedure with regard to the l_0- and l_∞-norms follows that outlined by Deb et al. [22]. Given two solutions s^1 and s^2 with corresponding norms l_0^1, l_∞^1, l_0^2, and l_∞^2, s^1 dominates s^2 if $l_0^1 < l_0^2$ and $l_\infty^1 \leq l_\infty^2$. If $l_0^1 < l_0^2$ but $l_\infty^1 > l_\infty^2$, then s^1 and s^2 are said to be non-dominated solutions. And finally, if s^1 and s^2 are both not adversarial, then the solution with the smaller loss function dominates the other.

Algorithm 1 Main Attack

Input: image x, population size N, max iterations G, probability of crossover p_c, probability of mutation p_m
Output: optimal set of adversarial examples

```
1: $P \leftarrow$ initialize_population($x$, $N$)
2: for $i \leftarrow 1$ to $G$ do
3:     parents $\leftarrow$ tournament_selection(N, P)
4:     for $k \leftarrow 1$ to number of parents / 2 do
5:         $par^1$, $par^2$ $\leftarrow$ parents
6:         $off^1$, $off^2$ $\leftarrow$ crossover($par^1$, $par^2$, $p_c$)
7:         $off^1$, $\leftarrow$ mutation($off^1$, $p_m$)
8:         $off^2$, $\leftarrow$ mutation($off^2$, $p_m$)
9:         $P \cup off^1 \cup off^2$
10:    end for
11:    $P_{0 \text{ to } z}$ $\leftarrow$ sort population into $z$ non-dominated levels
12:    remove solutions in lowest levels until size of population = N
13: end for
14: return $P_0$                                  ▷ return best pareto front
```

Each solution contains a set of values for each rgb channel in a pixel and each element in the set refers to a corresponding pixel in the input image. This set is referred to as *rgb* in Algorithms 2 and 5. Each element in *rgb* contains a tuple of 3 values referring to each rgb channel and these values can be -1, 0, or 1. Each solution also contains an attribute c that refers to the amount of perturbation applied to each pixel. The values in the set *rgb* are multiplied by the value of c and are combined with the input image x to form an adversarial image. If an element in *rgb* is (0,0,0), this means that a pixel will receive no change in the adversarial image, but if any of the numbers in the tuple are not zero then the pixel will receive a perturbation.

In the initialization step of our algorithm, N solutions are randomly generated to form the population P. Specifically, the number of pixels in *rgb* that will be nonzero is represented by the variable e in Algorithm 2, and this value is generated randomly during initialization. The locations, or indices, of these nonzero pixels in *rgb* are randomly sampled across the set. The value c is randomly sampled from real numbers between 0 and 1.

Algorithm 2 Initialize Population

Input: image x, population size N
Output: population of solutions

```
1:  $y \leftarrow$ number of pixels in $x$          ▷ ie. 1024
2:  $rgb \leftarrow (0,0,0)^y$ set
3:  $M \leftarrow \{0 \cdots y\}$
4:  for $n \leftarrow 1$ to $N$ do
5:      $e \leftarrow$ rand(0,$y$)
6:      $c \leftarrow$ rand(0,1)          ▷ perturbation amount
7:      $B \leftarrow U(M)^e$          ▷ uniformly sample $e$ indices
8:      $rgb_B \leftarrow$ rand(-1, 0, 1)$^e$
9:      $P \leftarrow A_B * c$
10: end for
11: return $P$
```

Within each generation, parents are selected via tournament selection as outlined in Algorithm 3. Solutions are selected for the tournament randomly and the winner is determined by the domination procedure described earlier. If two solutions are non-dominated then the solution with the superior crowding distance is selected. It is possible for the same solution to be selected as a parent multiple times within a generation, depending on how high its fitness is. The number of offspring created in each generation is equal to the value of N. Two offspring are created from each set of parents and each undergoes a crossover and a mutation operation.

Algorithm 3 Tournament Selection

Input: population size N, population P
Output: set of selected parents

```
1:  for $j \leftarrow 1$ to $N/2$ do          ▷ tournament selection
2:      $s^1$, $s^2 \leftarrow$ randomly choose 2 members of P
3:      if $s^1$ dominates $s^2$ then
4:          $par^1 \leftarrow s^1$
5:      else
6:          $par^1 \leftarrow s^2$
7:      end if
8:      $s^1$, $s^2 \leftarrow$ randomly choose 2 members of P
9:      if $s^1$ dominates $s^2$ then
10:         $par^2 \leftarrow s^1$
11:     else
12:         $par^2 \leftarrow s^2$
13:     end if
14:     parents $\leftarrow par^1, par^2$
15: end for
16: return parents
```

During the crossover operation, the offspring are first created as copies of each parent. Each offspring then receives a randomly sampled portion of the other parent's pixels that replace the rgb values in those pixel locations. The size of the portion of pixels a is determined by multiplying the probability of crossover p_c by the total number of pixels y in the image. The value of c for the offspring is also changed during this stage via simulated binary crossover [27]. Crossover is typically an operation applied to sequences such as a set of genes. Deb et al. proposed simulated binary crossover as a means of representing a continuous variable as a sequence so that a crossover-like operation can occur on it.

Algorithm 4 Crossover

Input: par^1, par^2, probability of crossover p_c
Output: 2 offspring

```
1: $y \leftarrow$ number of pixels in $par^1$
2: $M \leftarrow \{0 \cdots y\}$
3: $a \leftarrow$ round($y * p_c$)
4: $R^1 \leftarrow U(M)^a$          ▷ uniformly sample $a$ pixel locations
5: ${off^1}_{R^1} \leftarrow {par^2}_{R^1}$          ▷ replace the rgb values with parent's
   rgb values
6: $R^2 \leftarrow U(M)^a$
7: ${off^2}_{R^2} \leftarrow {par^1}_{R^2}$
8: ${off^1}_c$, ${off^2}_c \leftarrow$ simulated_binary_crossover(${par^1}_c$, ${par^2}_c$)
9: return $off^1$, $off^2$
```

During mutation, the locations of some of the nonzero rgb values in an offspring are moved within the image. The number of nonzero pixels that will change location d is determined by multiplying the probability of mutation p_m by the number of nonzero rgb values k. In Algorithm 5, these rgb values that will change pixel location are represented in the set B. The $k - d$ nonzero rgb values that will not change location are represented by the set A. The nonzero rgb values in set B will be moved to uniformly sampled pixel locations whose rgb values are all zero. The set C represents rgb values of (0,0,0) that will encompass pixels that are not represented by sets A or B. The value of c for the offspring will also undergo mutation via polynomial mutation. Similar to simulated binary crossover, polynomial mutation allows for a continuous variable to be represented as a sequence that can be mutated [28].

Algorithm 5 Mutation

Input: offspring *off*, probability of mutation p_m
Output: mutated offspring

```
1: $rgb \leftarrow off_{\text{rgb}}$
2: $rgb_{\text{nonzero}} \leftarrow rgb$ where $rgb \neq (0,0,0)$
3: $rgb_{\text{zero}} \leftarrow rgb$ where $rgb = (0,0,0)$
4: $y \leftarrow$ count($rgb$)
5: $k \leftarrow$ count($rgb_{\text{nonzero}}$)
6: $d \leftarrow$ round($k * p_m$)
7: $A \leftarrow U(rgb_{\text{nonzero}})^{\text{k-d}}$    ▷ sample unchanged pixels
8: $B \leftarrow U(rgb_{\text{zero}})^{\text{d}}$    ▷ pixels that will receive perturbations
9: $B \leftarrow$ rand(-1,0,1)    ▷ randomly assign rgb values
10: $C \leftarrow A \cup B$ / $rgb$    ▷ rgb values excluding sets A and B
11: $C \leftarrow (0,0,0)^{y - k - k + d}$    ▷ all rgb channels in C receive zero
12: $off^{mut} \leftarrow off$
13: $off^{mut}{}_{\text{rgb}} \leftarrow A \cup B \cup C$    ▷ combine sets with original pixel ordering
14: $off^{mut}{}_c \leftarrow$ polynomial_mutation($off_c$, $p_m$)
15: return $off^{mut}$
```

After the offspring are created, they are evaluated via the fitness function which includes the loss function, l_0-norm, and the l_∞-norm of the perturbed array. The loss function is found by combining the perturbed pixels in *rgb* with the input image x and inputting the resulting adversarial image into an image classifier model. The l_0-norm is calculated by counting the number of pixels with nonzero rgb values in *rgb*. Finally, the l_∞-norm is determined by the value of c of each solution.

The offspring are combined with the current population set P and are then sorted into non-dominated levels according to the convention in NSGA-II. This will result in a population that is greater than N, and the inferior solutions of the set will then be removed until the size of the population equals N again.

The above mentioned steps will repeat until the maximum number of iterations, or generations, is reached. The final output of the algorithm is the outermost non-dominated sorting level of P in the final generation. This resulting Pareto Front represents a set of perturbed rgb values that can be applied to the input image to cause misclassification. Each solution will generally have varying values for their l_0- and l_∞-norms.

IV. Experiment Setup

The conditions set for our experiment studying the Pareto Fronts of images are as follows. The algorithm described above will be run with a computational budget of 10,000 fitness evaluations. When the algorithm is run for a given test image, it will fully exhaust this budget and not terminate early if an adversarial solution is found. The number of iterations G that the algorithm will operate for on an image is calculated by dividing the computational budget by the size of the population N. For instance, if $N = 100$, then the algorithm would run for 100 iterations. The decision to fully exhaust the computational budget was made because of the assumption that more computation would be needed to optimize two objectives rather than one objective. Thus, this algorithm may require more computation than the single-objective adversarial attack algorithms found in the literature.

During the experiment, the parameter of N was set to 100. The large value for this parameter was chosen to increase the diversity of l_0- and l_∞-norm attributes in solutions. A greater number of solutions is thought to lead to more combinations of potentially optimal portions of these attributes, and therefore more diverse Pareto Fronts in the final output. The probability of crossover p_c was set to 0.1 and the probability of mutation p_m was set to 0.4. The tournament size during parent selection was set to 2 throughout the experiment. With the exception of N, these parameter values were chosen from following the experiment settings in Williams' work [7].

The experiment uses images from the CIFAR10 dataset created by Alex Krizhevsky [29]. This dataset contains small images that are 32 pixels wide and 32 pixels long, and contain 1024 pixels in total. It contains color images that are organized into 10 classes: airplane, automobile, bird, cat, deer, dog, ship, and truck. The dataset is a popular benchmark for image recognition models and adversarial attack algorithms. The simplicity of the 10 classes has made it useful for this experiment so that results from images of the same class can be aggregated together. For each neural network studied in the experiment, 100 images randomly sampled from each of the 10 classes were used as test images.

Three neural network models were included in the experiment to study the distinct vulnerabilities of each. These models were imported from the Robust Bench library [21], and include two defended models that were trained on adversarially perturbed images and one standard model that was trained on non-perturbed images. Details on the models used in the experiment are included in Table I.

TABLE I
Neural Networks Studied

Model ID	Accuracy	Adv. Accuracy	Architecture
Standard	94.78%	0.00%	WideResNet-28-10
Defended 1	92.16%	67.73%	WideResNet-28-10
Defended 2	83.34%	43.21%	PreActResNet-18

The model labeled as Defended 1 refers to a defended model created by Cui et al. [18] and Defended 2 refers to that created by Wong et al. [19]. At the time of this paper, the two models are ranked number 2 and number 65 respectively in Robust Bench's leaderboard for defended models. The Accuracy column in Table I refers to each model's accuracy on non-perturbed images. The Adv. Accuracy column, or adversarial accuracy, refers to the Robust Accuracy column on the leaderboard that displays the model's accuracy on perturbed images created by the AutoAttack algorithm [21].

Studying defended networks allows for the opportunity to evaluate the effectiveness of proposed defenses against adversarial attacks using multi-objective optimization. It should be noted that the accuracies of these models on non-perturbed images differ, and a test image was only included in the results if it was correctly classified by the neural network before perturbations were added to it.

V. Results

An overview of the results of our experiment are listed in Tables II-IV and in Figures 1-3. The tables depict summaries of the optimality of the Pareto Fronts for each class in the studied networks. The figures depict visualizations of the resulting Pareto Fronts for each class and their shapes. We will now explain the information displayed in these tables and figures, and will later describe the patterns observed in the standard model, and finally will describe the comparisons between the results of both the standard and defended models.

In the tables, the Min l_∞ column refers to the lowest l_∞ value achieved by the average Pareto Front for that class of images. This can also be visualized in the corresponding figure as the smallest x-coordinate reached by the curve. This value effectively represents the minimum amount of l_∞-norm perturbation required to cause misclassification in the image. Likewise, the Min l_0 column refers to the minimum amount of l_0-norm required to cause misclassification in the image, and can also be visualized in the corresponding figure. These two columns are further valuable because they can indicate a vulnerability to a particular norm relative to the other classes attacked on the network. On the other hand, a relatively higher value for a norm indicates that a class is more robust to that particular norm. The values for the l_∞- and l_0-norms in both the tables and the figures are normalized between 0 and 1. For instance, an l_∞ value of 0.1 refers to a perturbation of 10% of the pixel intensity range, which is effectively 10% of the maximum intensity value of 256 for 8-bit images. For the l_0-norm, a value of 0.1 refers to 10% of the 1024 pixels in a CIFAR10 image being perturbed.

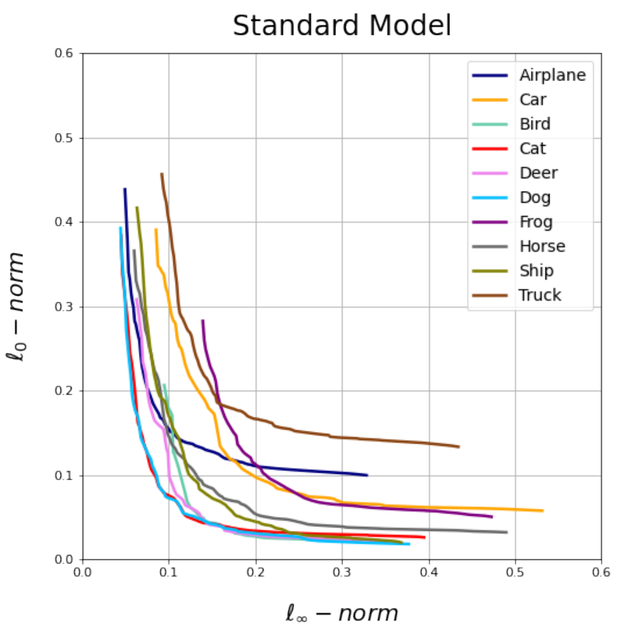

Fig. 1. Pareto Fronts From Standard Model

TABLE II
Summary of Standard Model

Class	Min l_∞	Min l_0	Hypervolume
Airplane	0.05	0.10	0.85
Automobile	0.09	0.06	0.84
Bird	0.09	0.02	0.88
Cat	0.04	0.03	0.92
Deer	0.06	0.02	0.91
Dog	0.04	0.02	0.93
Frog	0.14	0.05	0.81
Horse	0.06	0.03	0.89
Ship	0.06	0.02	0.90
Truck	0.09	0.13	0.77

TABLE III
Summary of Defended 1 Model

Class	Min l_∞	Min l_0	Hypervolume
Airplane	0.25	0.16	0.61
Automobile	0.24	0.19	0.60
Bird	0.27	0.17	0.60
Cat	0.14	0.05	0.81
Deer	0.15	0.06	0.79
Dog	0.17	0.13	0.72
Frog	0.31	0.18	0.54
Horse	0.22	0.14	0.64
Ship	0.24	0.12	0.65
Truck	0.29	0.26	0.51

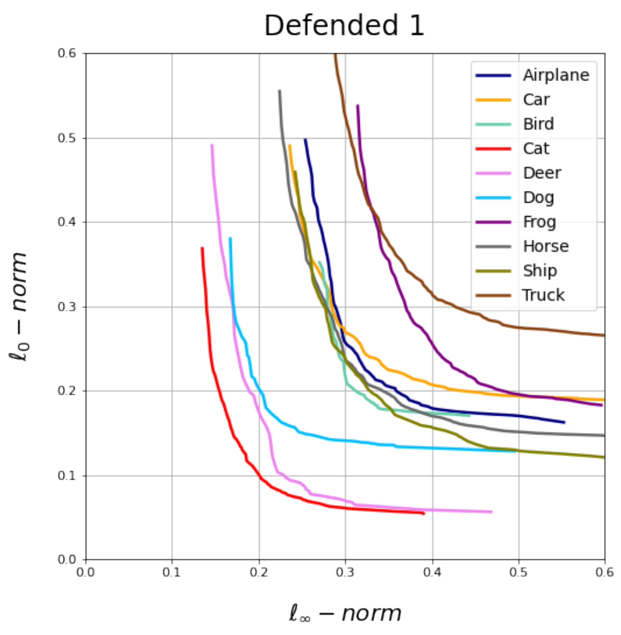

Fig. 2. Pareto Fronts From Defended 1 Model

TABLE IV
SUMMARY OF DEFENDED 2 MODEL

Class	Min l_∞	Min l_0	Hypervolume
Airplane	0.23	0.15	0.64
Automobile	0.22	0.18	0.62
Bird	0.19	0.07	0.73
Cat	0.14	0.07	0.77
Deer	0.12	0.07	0.81
Dog	0.17	0.10	0.73
Frog	0.52	0.19	0.39
Horse	0.23	0.12	0.65
Ship	0.22	0.13	0.66
Truck	0.28	0.22	0.54

The hypervolume column refers to the hypervolume value of the average Pareto Front for a given class. In multi-objective optimization, the indicator is commonly used as a measurement of the quality of convergence of a Pareto Front [30], [31]. It measures the volume of the dominated portion of the objective space [32] and can be thought of as the amount of two-dimensional space covered by the curve when there are two objectives. A higher hypervolume value for a class indicates that the particular class is overall more vulnerable to the adversarial attack of this algorithm than a class with a lower hypervolume value. Conversely, a lower hypervolume value indicates that the class is more robust to adversarial attacks.

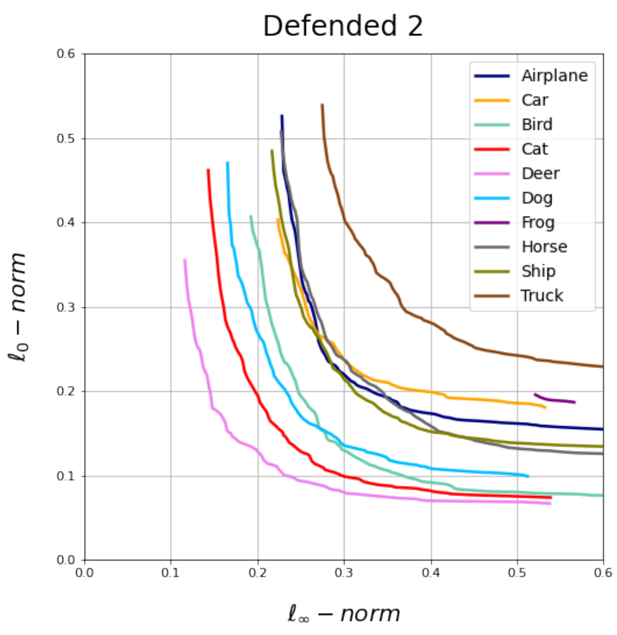

Fig. 3. Pareto Fronts From Defended 2 Model

By looking at the curves in the figures, one can discern visually which classes are more vulnerable or robust than others. While each Pareto curve for a class appears to be one continuous line, this is in fact not the case for the Pareto Front for an individual image. An image's resulting Pareto Front from the algorithm is a series of dots that generally resemble a convex curve with gaps between each dot. By using interpolation, a continuous line is created from the minimum x-coordinate to the maximum x-coordinate. An average of each line created from each image of the same class is then displayed as the curve shown in the figures. An example of this can be seen in the Pareto Fronts for the dog class in Figure 4, where each differently colored dot represents a Pareto solution from a different image.

This context is useful because some images have a significant amount of discontinuous space in their Pareto Fronts, and it should not be assumed that the curve of each class is continuous. An interesting research opportunity would be to study the discontinuous nature of these fronts individually from images and collectively as a class. The discontinuous nature of these fronts is not studied in this work because without additional, rigorous experimentation it is not possible to infer whether the gaps in the Pareto Fronts represent areas of invulnerability to misclassification or if the algorithm simply could not find optimal solutions that in reality do exist in that region of the objective space. This is one reason why assessing quality of Pareto Fronts is a challenging topic in multi-objective research [33], [34]. For sake of simplicity in this experiment, the Pareto curves are therefore represented as

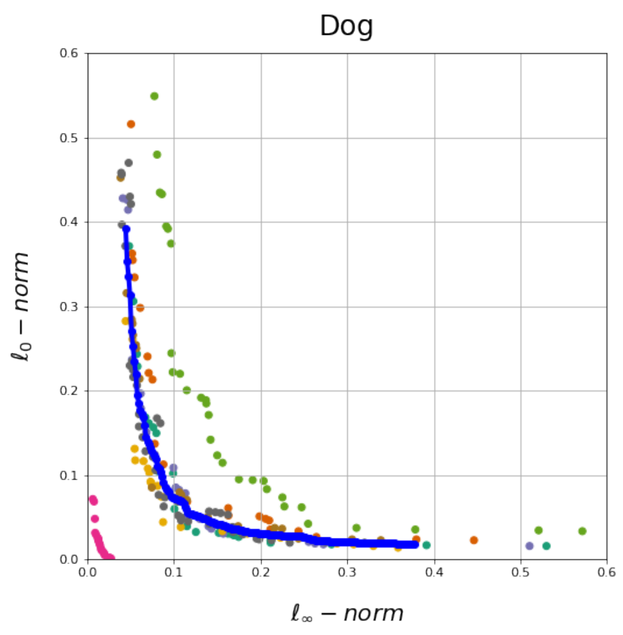

Fig. 4. Pareto Front for dog class created as an average of interpolated curves from individual Pareto Fronts

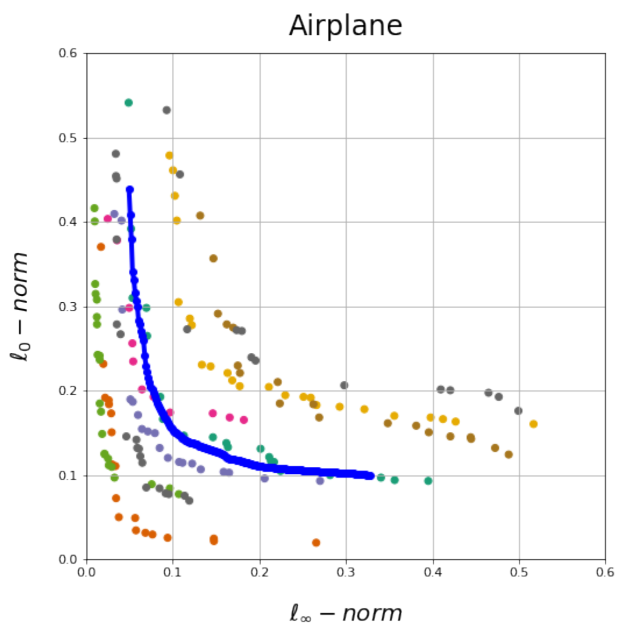

Fig. 5. Pareto Front for airplane class created as an average of interpolated curves from individual Pareto Fronts

continuous lines.

As a note, the results on the frog class in the Defended 2 model are outliers. Because the Pareto solutions for each frog image in this model's experiment are sparsely placed in several, distant locations in the objective space, the approximate interpolation is a very short line segment. When comparing classes between Defended 2 and other models, the frog class will be excluded from discussion.

A. *Patterns of Classes in Standard Model*

The results of our experiment show that the vulnerabilities of a neural network can vary depending on the class of image. In the standard WideResNet model that did not receive adversarial training, we will now explore some observed patterns among the classes.

Some classes, such as cat and dog, were particularly more vulnerable than the other classes. As seen in Figure 4, many of the optimal solutions are located relatively close to the x- and y-axes. This is also reflected in the dog class having the highest hypervolume value of 0.93.

Other classes showed vulnerability to one norm while showing robustness to another norm. An example of this is the airplane class displayed in Figure 5. Compared to the dog class in Figure 4, the average curve appears to be shifted upwards. This suggests that while the airplane class is still significantly vulnerable to l_∞-norm perturbations, it is more robust to l_0-norm perturbations because of the fewer dots that have y-coordinates less than 0.1. Similarly, the frog class appears to be more robust to l_∞-norm perturbations but still very vulnerable to l_0-norm perturbations.

A contrast to these patterns is seen in the truck class, which appears to be more robust to both l_∞- and l_0-norm perturbations. This is also shown by its having the lowest hypervolume value of 0.77.

B. *Comparing to Defended Models*

By comparing Figure 1 with Figures 2 and 3, one can observe that that each of the classes in the defended models are more robust than those in the standard model. These differences in robustness are quantified in Tables V and VI. In the l_∞ column, the difference in l_∞ robustness is calculated from finding the average difference in x-coordinates for each class. Likewise, the l_0 column is calculated from the average difference in y-coordinates. The Hypervolume column shows the indicator's difference between the two models.

By observing the differences shown in these tables, one could learn of the amount of robustness to each norm gained by a defended network. For defended networks that undergo adversarial training, one would expect that this amount would be strictly related to the l_p norm applied to each image during training. However, the Defended 2 model used an l_∞ perturbation of 8/255 (or, 0.03 as a normalized value) [19], but the added robustness in our analysis appears to be more than that for each of the classes. Further research would be needed to understand why the added robustness is greater than the perturbation amount during training.

Furthermore, these differences can be used to evaluate the quality of the defense of a network. For instance, in the standard network the deer class is very vulnerable to l_0-norm perturbations but less vulnerable to the l_∞-norm. As

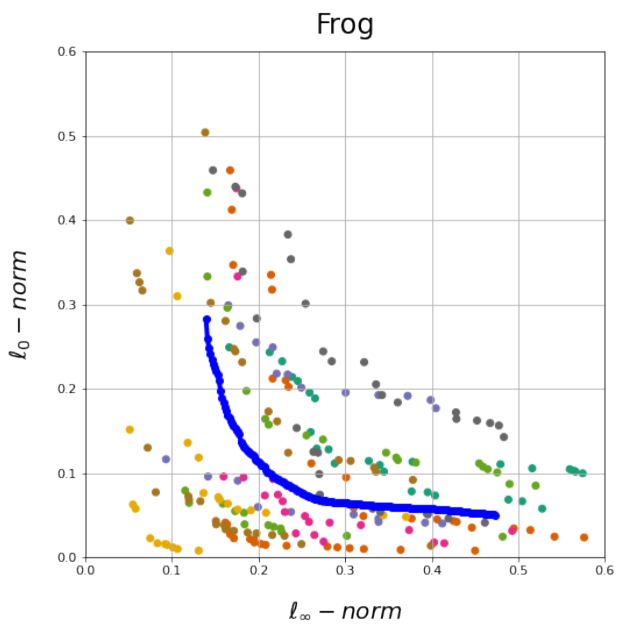

Fig. 6. Pareto Front for frog class created as an average of interpolated curves from individual Pareto Fronts

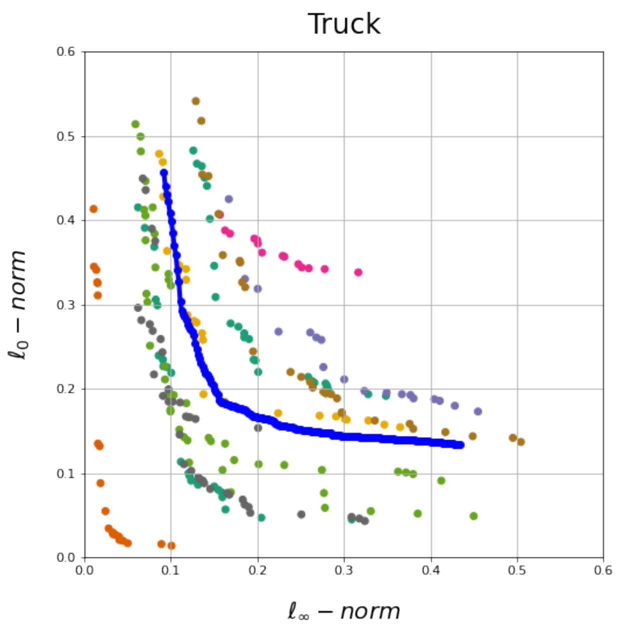

Fig. 7. Pareto Front for truck class created as an average of interpolated curves from individual Pareto Fronts

indicated by the in Table VI, the deer class on average receives 0.12 additional robustness to the l_∞-norm, but only 0.05 to the robustness of the l_0-norm. Thus, this defense offers less protection to its more vulnerable norm and an attacker could be motivated by this information to use a more sparse attack to cause misclassification.

The respective adversarial accuracies for Defended 1 and Defended 2 listed in Table I suggest that Defended 1 is significantly more robust than Defended 2. We can use the results from this experiment to also compare the general robustness between these two defended networks and to verify if Defended 1's robustness is superior to that of Defended 2. In Tables III and IV, the hypervolume indicator seems to validate this conclusion as Defended 1 has lower hypervolume values than Defended 2 in 8 of the 10 classes. Defended 1's adversarial training appears to be more balanced than that of Defended 2, as 9 of the 10 classes in Defended 2 have significantly higher l_∞-norm differences than l_0-norm differences, as shown in Table VI. As shown in Table V, only 6 of the 10 classes have higher l_∞-norm differences than l_0-norm differences. When using a Wilcoxon signed rank test [35] to compare the two sets of hypervolumes from the models (excluding the frog class), it appears however that the differences in hypervolumes are not significant enough to show there is a true difference in robustness among the 9 classes. The resulting p-value of the Wilcoxon test is only 0.074, which is slightly greater than the typical p-value threshold of 0.05 needed to reject the null hypothesis that the two hypervolumes are from the same population. While bearing in mind that the p-value is not far from 0.05, this does suggest that Defended 1 may not be significantly better than Defended 2 judging by the hypervolumes of their classes' average Pareto Fronts. However, as mentioned previously, Defended 2 may be more vulnerable to sparse l_0-norm attacks.

TABLE V
DIFFERENCES BETWEEN STANDARD AND DEFENDED 1

Class	l_∞	l_0	**Hypervolume**
Airplane	0.21	0.08	0.24
Automobile	0.13	0.13	0.24
Bird	0.15	0.16	0.28
Cat	0.04	0.04	0.11
Deer	0.10	0.06	0.12
Dog	0.12	0.10	0.21
Frog	0.15	0.16	0.26
Horse	0.16	0.13	0.25
Ship	0.21	0.10	0.26
Truck	0.21	0.14	0.27

TABLE VI
DIFFERENCES BETWEEN STANDARD AND DEFENDED 2

Class	l_∞	l_0	**Hypervolume**
Airplane	0.23	0.06	0.21
Automobile	0.07	0.12	0.22
Bird	0.23	0.08	0.15
Cat	0.12	0.07	0.15
Deer	0.12	0.05	0.10
Dog	0.13	0.10	0.19
Frog	0.24	0.10	0.42
Horse	0.18	0.11	0.24
Ship	0.21	0.11	0.24
Truck	0.21	0.10	0.23

VI. CONCLUSION

In this paper, we have introduced a multi-objective approach to adversarial attack algorithms and a novel way of studying models' vulnerabilities to adversarial examples via Pareto Fronts. We have also detailed the steps of the proposed attack algorithm as a framework that can be used for future algorithms that use multi-objective evolutionary optimization. We have also discussed the results of our experiment for the algorithm on 3 ResNet models. Our analysis shows that the vulnerabilities of a model vary depending on the class of image being attacked. Our analysis also provides a framework for evaluating the robustness of defended models.

We believe this work represents an unexplored area within adversarial attack research and that many potential directions exist for future research. One such opportunity explores how this algorithm scales to larger images from the ImageNet dataset. Another includes a deeper study into the effects of tuning the parameters used in this algorithm. Finally, the experiment in this work focused on standard and defended models in the ResNet family, but an interesting opportunity lies in studying the algorithm's results on other state-of-the-art architectures such as MobileNetV2 and ConvNeXt [36].

REFERENCES

[1] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, 2019. [Online]. Available: https://doi.org/10.1109/TEVC.2019.2890858

[2] L. Sun, M. Tan, and Z. Zhou, "A survey of practical adversarial example attacks," *Cybersecur.*, vol. 1, no. 1, p. 9, 2018. [Online]. Available: https://doi.org/10.1186/s42400-018-0012-9

[3] M. Alzantot, Y. Sharma, S. Chakraborty, and M. B. Srivastava, "Genattack: Practical black-box attacks with gradient-free optimization," *CoRR*, vol. abs/1805.11090, 2018. [Online]. Available: http://arxiv.org/abs/1805.11090

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[5] H. Qiu, L. L. Custode, and G. Iacca, "Black-box adversarial attacks using evolution strategies," *CoRR*, vol. abs/2104.15064, 2021. [Online]. Available: https://arxiv.org/abs/2104.15064

[6] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: a query-efficient black-box adversarial attack via random search," *CoRR*, vol. abs/1912.00049, 2019. [Online]. Available: http://arxiv.org/abs/1912.00049

[7] P. Williams, K. Li, and G. Min, "Sparse adversarial attack via bi-objective optimization," in *Evolutionary Multi-Criterion Optimization*, M. Emmerich, A. Deutz, H. Wang, A. V. Kononova, B. Naujoks, K. Li, K. Miettinen, and I. Yevseyeva, Eds. Cham: Springer Nature Switzerland, 2023, pp. 118–133.

[8] T. Suzuki, S. Takeshita, and S. Ono, "Adversarial example generation using evolutionary multi-objective optimization," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 2136–2144.

[9] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015.

[10] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016. [Online]. Available: http://arxiv.org/abs/1608.04644

[11] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: Elastic-net attacks to deep neural networks via adversarial examples," 2018.

[12] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh, "ZOO: zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*, B. Thuraisingham, B. Biggio, D. M. Freeman, B. Miller, and A. Sinha, Eds. ACM, 2017, pp. 15–26. [Online]. Available: https://doi.org/10.1145/3128572.3140448

[13] F. Croce, M. Andriushchenko, N. D. Singh, N. Flammarion, and M. Hein, "Sparse-rs: a versatile framework for query-efficient sparse black-box adversarial attacks," *CoRR*, vol. abs/2006.12834, 2020. [Online]. Available: https://arxiv.org/abs/2006.12834

[14] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," 2018.

[15] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," 2018.

[16] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," 2017.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019.

[18] J. Cui, Z. Tian, Z. Zhong, X. Qi, B. Yu, and H. Zhang, "Decoupled kullback-leibler divergence loss," 2023.

[19] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," 2020.

[20] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian, and T. Mann, "Improving robustness using generated data," 2021.

[21] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," 2021.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[23] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, 1999. [Online]. Available: https://doi.org/10.1162/evco.1999.7.3.205

[24] V. De Buck, P. Nimmegeers, I. Hashem, C. A. Muñoz López, and J. Van Impe, "Exploiting trade-off criteria to improve the efficiency of genetic multi-objective optimisation algorithms," *Frontiers in Chemical Engineering*, vol. 3, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fceng.2021.582123

[25] S. Gass and T. Saaty, "The computational algorithm for the parametric objective function," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 39–45, 1955. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020106

[26] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007. [Online]. Available: https://doi.org/10.1109/TEVC.2007.892759

[27] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[28] K. Deb and D. Deb, “Analysing mutation schemes for real-parameter genetic algorithms,” *International Journal of Artificial Intelligence and Soft Computing*, vol. 4, no. 1, pp. 1–28, 2014.

[29] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[30] J. Bader, K. Deb, and E. Zitzler, “Faster hypervolume-based search using monte carlo sampling,” in *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, M. Ehrgott, B. Naujoks, T. J. Stewart, and J. Wallenius, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 313–326.

[31] N. Beume, C. M. Fonseca, M. Lopez-Ibanez, L. Paquete, and J. Vahrenhold, “On the complexity of computing the hypervolume indicator,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1075–1082, 2009.

[32] T. Friedrich. [Online]. Available: https://hpi.de/friedrich/research/the-hypervolume-indicator.html#:~:text=A%20measure%20that%20has%20been,feature%20of%20strict%20Pareto%20compliance.

[33] Y. Hua, Q. Liu, K. Hao, and Y. Jin, “A survey of evolutionary algorithms for multi-objective optimization problems with irregular pareto fronts,” *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 303–318, 2021.

[34] Y. Hua, Y. Jin, and K. Hao, “A clustering-based adaptive evolutionary algorithm for multiobjective optimization with irregular pareto fronts,” *IEEE Transactions on Cybernetics*, vol. 49, no. 7, pp. 2758–2770, 2019.

[35] J. W. Pratt, “Remarks on zeros and ties in the wilcoxon signed rank procedures,” *Journal of the American Statistical Association*, vol. 54, no. 287, pp. 655–667, 1959. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1959.10501526

[36] K. Team, “Keras documentation: Keras applications.” [Online]. Available: https://keras.io/api/applications/