Comparing Machine Learning and Multi-Objective Optimization Algorithms for the Design of Lightweight Mechanical Gears

Abigail Kwon

Received July 19, 2025
Accepted August 14, 2025
Electronic access October 15, 2025

The design of lightweight mechanical structures and machines, has the potential to help reduce the environmental impact of our industrialized society. Lightweight mechanical gears hold promise, because they are common components in many machines. However, testing mechanical designs can be costly and time-consuming, which can discourage innovation. Modern computational technologies offer a way to improve designs through faster, lower-cost methods. In particular, data-driven design approaches using Machine Learning (ML) models and Multi-Objective Optimization (MOO) processes, have shown potential. Effective gear design requires exploring geometric and material variations, because "real" objects are most fully characterized by both continuous numerical information (geometric dimensions) and discreet categorical information (materials). However, there is very little current literature on data-driven design research with gears using this kind of mixed data. This study attempts work in this gap by investigating the performance of various ML model and MOO process combinations for designing lightweight mechanical gears with mixed data. These combinations led to six algorithms. AutoGluon, Keras Neural Networks, and Random Forest ML models were used, due to their ability to work with mixed data. A random mixed dataset for this study was generated using Python and the CAD software package SolidWorks. This dataset was then used with the six algorithms, and the performance of each algorithm was compared. The results of this study show that an algorithm using multiple AutoGluon models and multiple MOO processes performed the best, while an algorithm using a single AutoGluon model with a single MOO process performed second best. AutoGluon is an example of an automated stacked-ensemble ML model, that does not require the manual tuning of hyperparameters that many other ML models require. The ability to get high performance outcomes with mixed data, without requiring expert ML model tuning knowledge, may encourage more designers and engineers to adopt data-driven design approaches for producing a wider array of lightweight and efficient mechanical parts and machines. Further research with other automated ML models and other mechanical objects would also be valuable.

1 Introduction

Making better use of Earth's limited natural resources is a goal that appropriately receives a lot of attention. Designing more efficient and lightweight mechanical structures and machines will help us move towards that goal. However, a design process that relies mostly on physical manufacturing and testing of various mechanical designs, can be very expensive and time-consuming[1 2]. A good way to address this issue is to use a data-driven design approach that uses a combination of surrogate Machine Learning (ML) models and genetic multi-objective optimization (MOO) algorithms to help search for optimal designs[3 4]. This up-front computational search, added to traditional design approaches, reduces the time and resources needed for physical and other formal testing. Data-driven design is promising and growing in use in many industries including automotive and aerospace[3 5 6].

To reduce our modern society's environmental impact, many in the automotive and aerospace industries engage in "lightweighting." Lightweighting involves reducing the mass of parts and vehicles while maintaining other structural, economic, or environmental requirements[1 7 8 9 10]. A lot of work reducing the mass of larger machine components has already been done[7]. But much less work has been done improving smaller components such as gears[11]. Lightweight mechanical gears are a good target for additional work, because they are common components used in many kinds of machines from motors to power transmission systems in cars and airplanes[8]. Improved lightweight gear designs have a very broad potential impact through the global automotive and aerospace industries. For example, it is estimated that lighter gearboxes in aircraft can result in up to 15% of the total mass savings, which would lead to significant fuel savings[12]. Designing gears can be challenging though, because gears are often used in high performance environments with specific strength, weight, and durability requirements[11]. But the many potential benefits of lightweight gears, including reduced weight and reduced energy consumption of many different types of machines, makes research on their design worthwhile[8].

Typical approaches to creating lighter gears include changing

© The National High School Journal of Science 2025

the geometry by varying the web and rim thickness and cutting out sections of the body (see Figure 1).

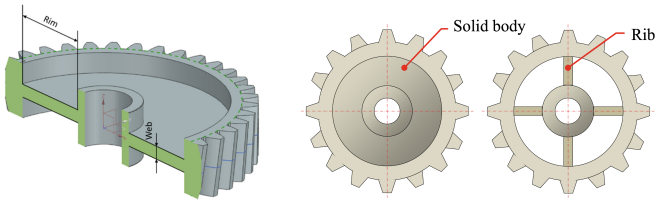


Fig. 1 Varying Web and Rim Thickness[8] and Body Cutouts[13]

The shape and number of cutouts can vary, leading to different performance characteristics. Examples of lightweight gears from research and industry can be seen in Figure 2[11 14 15].

Fig. 2 Sample Lightweight Gears

Other approaches include changing the materials of gears, since materials have varying physical characteristics[1 8]. Lower density materials are lighter, but also not as strong[7]. Some alloys are more brittle than others, depending on for example, aluminum content[9]. Some materials like copper are more corrosion resistant, making them more durable because of less degradation from lubricants[16]. Overall, work with materials can be challenging because of the interaction between material properties and other parameters[1].

As noted above, physically testing mechanical designs can be expensive and slow. Manufacturing a variety of mechanical gears is labor intensive, and the cost of materials can also be very high. Luckily, modern computational technologies offer a way to improve designs through faster, lower-cost methods.

1.1 Computer-Aided Design and Finite Element Analysis

Computer-aided design (CAD) tools and computer-aided manufacturing (CAM) tools, can be used to design and manufacture physical products[17]. These types of tools help by making it easier to create designs quickly, and easier to make changes to those designs. These tools can also be used to predict how well a product will perform in real life. CAD tools for example, can predict how much force a product can withstand before breaking by running simulations using finite element analysis (FEA). FEA results can be highly accurate compared to actual physical experiments, with reports ranging from 90% accuracy in an aerospace example[18] to 98% accuracy in an automotive example[19]. Finding product limitations with CAD simulations is faster and less expensive than building and testing numerous physical models[2 8]. CAD FEA does not completely replace physical testing in the design process, because it is understood to be an approximation. It serves as an important part of the overall process to significantly reduce the amount of physical testing required[20]. The use of FEA has been adopted by a wide range of industries including aerospace, automotive, architecture, and biomedical engineering[18 19 21]. Therefore, using CAD tools can lead to the faster design of products at a lower cost. However, because of the computational requirements of FEA simulations, exploring a wide range of possible designs can still be very time-consuming. A single simulation can take several minutes or even several hours[22]. This is faster than physical construction and testing, but still very slow.

1.2 Data-Driven Design

Since high-fidelity FEA is very time consuming and requires a lot of computational resources[23], researchers continue to search for technologies that enable the faster search of designs (sometimes called “rapid tools” or “decision-support systems”), as a step before high-fidelity FEA[21 23 24]. Advancements in artificial intelligence are providing some of these tools[4 5 21]. Examples include more powerful machine learning (ML) models and multi-objective optimization (MOO) algorithms[4 5 6]. These new technologies enable “data-driven” design practices through the use of surrogate ML models[4 5 21].

Data-driven design involves training ML models on design datasets. These design datasets can include data from existing physical product designs, physical experiments, or CAD designs[3 4]. The design data for training needs to include both mechanical parameters for input (including geometric dimensions) and desired performance measures for output (like mass and strength). The ML models trained on this design data, can then serve as “surrogate” models that can predict product performance given different parameter inputs[4 21 25]. To support the search for new designs, multi-objective optimization algorithms can be used with the ML models. Optimization algorithms can check thousands of design variations quickly using these surrogate ML models[3 4]. Using ML models is much faster than checking design variations with CAD FEA simulations. Surrogate ML models can provide accurate predicted output values in seconds, while a single FEA simulation can take several minutes or even several hours[4 22 25]. Therefore, an ML-MOO process can identify a range of where good design options may exist, very quickly. Data-driven design, like other “rapid tools”, does not replace FEA or physical testing in the design process. It serves as a time-saving step before FEA, to reduce the total time involved in the overall design process[4 21 24].

Described more precisely, the multi-objective optimization process, leads to finding sets of possible design solutions that

© The National High School Journal of Science 2025

are optimal, given various desired performance targets. These optimal solutions are called "non-dominated." For a solution to be non-dominated, there must be no other feasible solution that is better than that solution for every performance objective. These non-dominated solutions, are also called Pareto-optimal solutions[26]. When placed on a graph together, these Pareto-optimal solutions form a "Pareto front." An example Pareto front is shown in Figure 3[8].

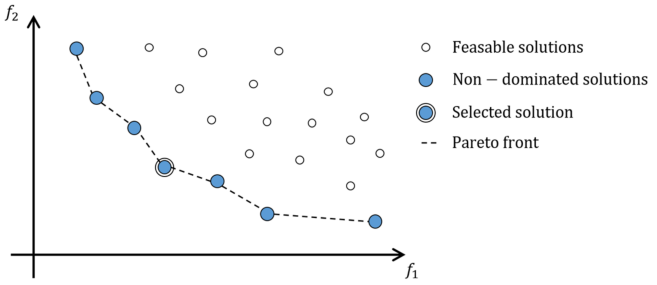


Fig. 3 Example of a Pareto front

Pareto fronts are useful to designers and engineers, because they provide an outline of where optimal solutions exist. For example, if in Figure 3 the x-axis was "strength" and the y-axis was "mass", a designer can use a Pareto front to determine the lowest possible mass something can have, given a certain strength requirement. A designer can also do the reverse, and choose first a target mass, and then see what the best possible strength could be[8]. Data-driven design processes using surrogate ML models and multi-objective optimization, can automate and quickly complete repetitive and time-consuming tasks. This greatly speeds up the overall design process.

There are many existing studies that have explored the use ML models and optimization algorithms in their work. For example, Regenwetter, Weaver, and Ahmed used the data-driven design approach to explore the design of bicycle frames[3]. They used an AutoML model and genetic optimization to generate bicycle frame designs with optimal performance characteristics. Stabile, Ballo, Gobbi, and Previati also used the data-driven design approach to generate wheel designs for light-weight electric vehicles[27]. They used neural networks and multi-objective optimization to minimize the mass while maintaining structural integrity, meeting required safety standards. Chen, Chui, and Fuge used data-driven design to create aerodynamic airfoils[28]. Their approach led to realistic and compact airfoil shapes and reduced computational cost by an order of magnitude. Looking at the design of smaller parts, Ramadani et al. used data-driven design to explore ways to reduce gear vibration and weight[13]. Their work employed optimization strategies, and focused on the structural topology of gears and how that affected the performance of gears. Urbas, Zorko, and Vukasinovic studied the use of ML models trained with FEA data to support the design of gears[23]. Their ML models served as faster surrogate models that computed results that on average only deviated 3.4% from the more accurate FEA computations.

There are actually several studies that combine multi-objective optimization with FEA or analytic methods. For example, Barbieri, Bonori, and Pellicano used genetic multi-objective optimization and FEA to reduce vibration and noise in spur gears[29]. Their work modified the geometry of the spur gear tooth and used FEA with MOO processes to find optimal designs. Miler, Zezilj, Loncar, and Vuckovic used genetic multi-objective optimization and analytic equations to optimize gear pair volume and efficiency[30].

1.2.1 Mixed Data

Research on lightweight mechanical objects points to the importance of considering both object geometry and materials[1 9 10]. Being able to explore designs while varying a full range of information (dimensions and materials) is better than exploring designs varying just a partial set of information[31]. Research specifically on gear design reinforces this importance[1 11 32]. There are several studies using FEA and MOO that worked to optimize designs by varying both dimensions and materials. Sokolov[33] studied gear design optimization while considering two materials. Pillai et al.[34] and Xin et al.[35] each studied gear design optimization with three materials. Even studies using just one material, note the importance of considering multiple materials in future work[8 32].

With data-driven design, surrogate ML models are used instead of FEA. This leads to the need for an ML-MOO process that can work with two types of information: continuous numerical (for geometric dimensions) and discreet categorical (for material type). That is, the optimization process must be able to work with "mixed data" or "mixed variables"[36 37]. This presents a challenge because many ML models do not work well with mixed data.

A systematic search of the literature reveals there is little current research on the use of mixed data with ML-MOO processes for the design of mechanical gears specifically. However, there does exist research with mixed data on other engineering objectives. For example, Li et al.[38] studied lightweight automobile engine hood design with mixed variables. Saves et al.[39] used mixed variables to study optimization of aircraft designs. Bartoli et al.[40] used mixed variables to study the optimization of chemical systems. But no studies were found specifically for using ML-MOO processes with mixed variables for optimizing gear design. This is likely the case because data-driven design with ML-MOO is a newer approach than FEA-MOO. As noted above, there are several studies using FEA-MOO for gear optimization with geometric dimensions and materials. It is valuable to take advantage of faster ML-MOO approaches to also address gear optimization with geometric dimensions and materials. A larger design space can be searched because of the significant increase

in speed with ML-MOO. More research is needed in this area, because this can enhance the gear design process even further.

Mixed data use in data-driven design can be improved by trying better sets of ML models and by trying different approaches to generating Pareto-optimal sets of results. There are some ML models that can handle mixed data fairly well. Three of the most promising types of ML models for this are Decision Trees[25,41], Neural Networks[42,43], and automated ensemble ML models[3,44].

1.2.2 Decision Trees

Although many ML models don't work well with mixed data, Decision Trees can handle mixed data well[25,41]. Decision Trees work by creating a tree-like structure representing answers to questions. An example decision tree is shown in Figure 4. The starting point is the root node, the secondary questions are decision nodes, and the final outcomes are leaf nodes. Because of the question-and-answer structure, both numerical and categorical data can easily be used at the same time during training. Decision Trees have been used in many applications including medicine, cybersecurity, and banking[41,45]. A variation called Random Forest involves the use of multiple Decision Trees working together. Random Forest tends to provide better performance than a single Decision Tree, and was therefore used in this study.

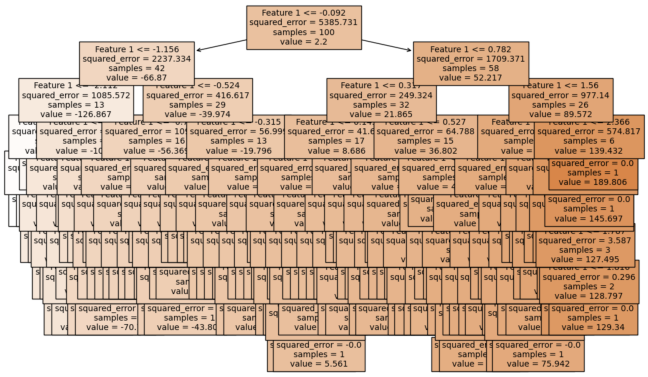


Fig. 4 Example of a Decision Tree

1.2.3 Neural Networks

Another type of ML model that can handle mixed data well is Neural Networks. Neural Networks are modeled after the human brain. The human brain is thought to consist of a large network of neurons, so in similar fashion, computer Neural Networks simulate a large network of connected nodes, with "input" nodes, "output" nodes, and several "hidden" layers of nodes in between. Each node can be connected to several other nodes and each connection is given a numerical "weight". These "weights" can be adjusted during training to try to yield better outputs. An example neural network is shown in Figure 5. The Keras implementation of Neural Networks is viewed as being especially effective and popular[42,43]. Neural Networks have been used in many applications including medicine and agriculture[46,47]. The use of the Keras implementation Neural Networks has led to high identification accuracy while using mixed numerical, categorical, and image data.

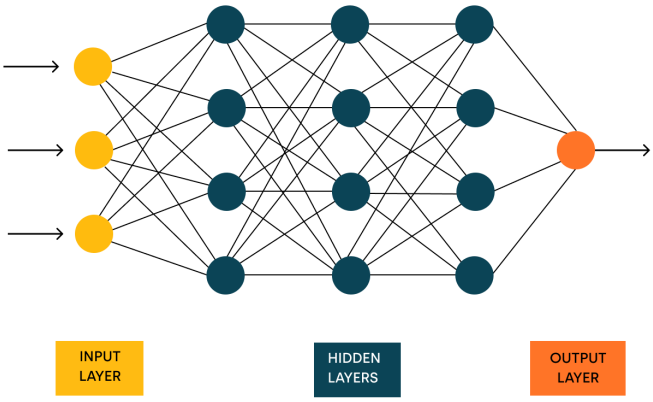


Fig. 5 Example of a Neural Network[48]

1.2.4 Automated ML

The newest type of ML model is an automated machine learning model (AutoML). Traditional approaches for selecting an ML model include a combination of intuition and trial-and-error[3,49]. Once selected, ML models also require hyperparameter tuning, also known as hyperparameter optimization (HPO)[50]. While effective HPO strategies like Bayesian optimization exist[50], some of the best modern MLs required manual design by ML-experts involving trial-and-error, which shows that even experts require substantial resources and time to create high performing models[51]. As ML techniques continue to grow in sophistication, it can be difficult for even ML experts to keep up with the latest work on selection and optimization. Learning all the best practices for selecting and using ML models can be time-consuming[52]. For those who are not experts, the barriers to learning and applying correct practices are even higher.

Yao et al. identify the three main goals of AutoML as (a) better ML performance, (b) no assistance from humans, and (c) lower computational budgets[44]. By automating a lot of the time-consuming work often done by ML experts, AutoML is supposed to enable non-ML-expert users to build machine learning applications quickly and automatically without extensive knowledge of machine learning and statistics[3,49]. AutoML should even help ML experts by automating tedious HPO tasks to save time. Furthermore, some current AutoML implementations have been shown to rival and even outperform human ML experts[53]. Though it is a newer technology, AutoML has been used in engine design optimization[54], quadcopter structural

© The National High School Journal of Science 2025

optimization[55], and aggregate concrete's compressive strength prediction[56].

Some AutoML models function by automatically trying several traditional ML models with different hyperparameters and choosing the best set of promising models. One advanced AutoML model called AutoGluon uses a "layer-stack ensembling approach", which combines the use of several ML models together to produce one higher-performing ensemble model[57]. AutoGluon has been used to help with many design and prediction tasks in mechanical engineering and medicine[3 58]. AutoGluon has been shown to provide better prediction accuracy in some tests than traditional ML models including K-Nearest Neighbors, Support Vector Machine, and Neural Networks[3] as well as other AutoML frameworks like Auto-WEKA, auto-sklearn, and TPOT[57].

1.3 Optimization Approaches

The usual approach is to use multi-objective optimization with one trained ML model to generate one Pareto optimal set of design solutions (see Figure 6). However, if one trained ML model alone does not perform well with all the mixed data, another approach is to separate the data and have ML models learn from the separated data. Then separate optimal design results are generated (see Figure 7). Afterwards, these separate results are combined to create one Pareto optimal set of results[27].

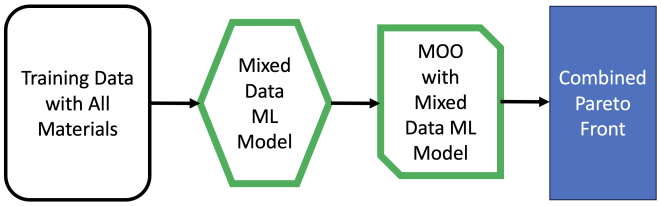


Fig. 6 Optimization with One ML Model Using Mixed Data

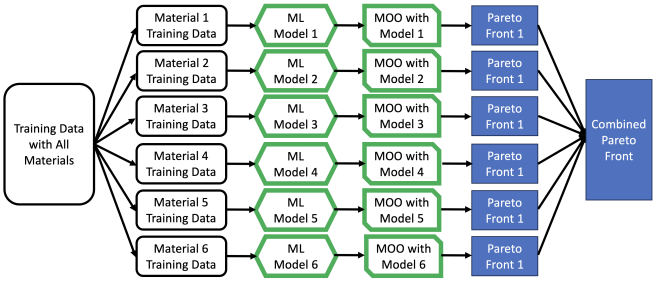


Fig. 7 Optimization with Multiple ML Models Using Separated Data

1.4 Methodology Overview

In this study, an experiment was performed where different algorithms for generating Pareto-optimal designs were compared. Three ML models were used in this experiment, namely AutoGluon, Keras Neural Networks, and Random Forest. Two optimization approaches were also used. The combination of different ML models and optimization approaches led to six different algorithms for working with mixed data (see Table 1). These algorithms were the independent variable. The Pareto-optimal design results (or Pareto fronts) generated by each of the algorithms, were the dependent variable. The quality of the design results, as measured by several performance indicators for Pareto fronts, determined which algorithms were the most effective.

The full process is shown in Figure 8. The first main set of steps is for creating a training dataset for the surrogate ML models. The second set of steps involves running the experiment to generate Pareto fronts for each of the algorithms. Figures 6 and 7 show more of the details that occur within this part of the process. The third set of steps is for adding FEA computed outputs to the Pareto front design datasets. The final set of steps is for computing performance indicators for the Pareto fronts, and comparing each of the algorithms based on these performance indicators.

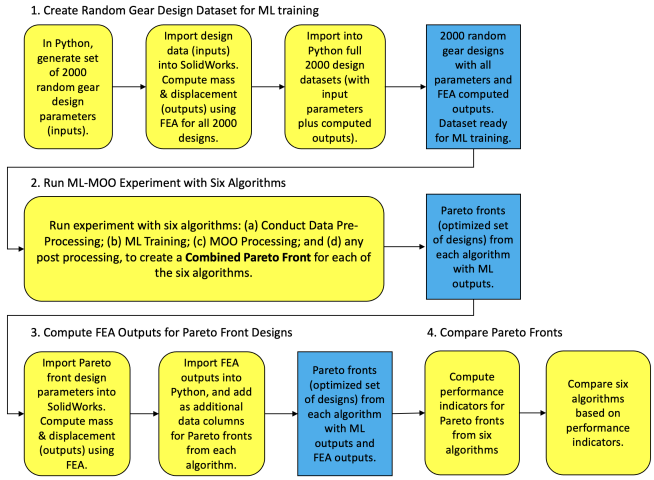


Fig. 8 Full Workflow Process

Knowing which algorithms are best for creating Pareto-optimal designs directly from mixed data, is very useful for designers, because being able to explore designs while varying a full range of information (dimensions and materials) is better than exploring designs varying just a partial set of information[8 9 10]. The results of this study can add to the body of scientific knowledge on working effectively with mixed data to design mechanical objects using inexpensive and fast, data-driven design methods.

Table 1 Combination of MOO Methods and ML Models for Different Algorithms

	One MOO using one ML Model leading to one Pareto Front	Multiple MOOs each with separate ML Models leading to multiple Pareto Fronts for merging into one Pareto Front
AutoGluon	Algorithm 1	Algorithm 4
Keras Neural Networks	Algorithm 2	Algorithm 5
Random Forest	Algorithm 3	Algorithm 6

2 Methods

This is an experimental study of six algorithms for generating Pareto-optimal designs of lightweight mechanical gears from mixed data. Three ML models and two optimization approaches were used, resulting in six total algorithms, as shown in Table 1.

To run this experiment, first, a training dataset for the ML models was generated. Using Python running in Visual Studio Code, 2000 random designs for gears with machined webs were generated. The same eight-opening web layout was used, with randomized values for thickness, inner radius, outer radius, inner angle, and materials. Base parameters for the gear included a 5 cm outer radius, 1 cm bore (central hole), 3.5 cm maximum rim, 2 cm maximum web thickness, and a 20° gear pressure angle (the most widely used standard).

Commercially manufactured gears come in a very wide range of sizes. Spur gears ranging in size from 1 cm to 10 cm are readily available for order by individuals online[59]. A spur gear with a radius of 5 cm was chosen for this experiment. Dimensional constraints were applied in Python to prevent the creation of physically invalid designs. For example, the outer radius of the cutout was set to always be greater than the inner radius, and the outer radius was limited so it would never be greater than the radius of the gear's body. Similarly, the inner cutout angle was constrained so that it would never lead to cutouts overlapping. A check of the distribution of randomized values is shown in Figure 9. The histograms show the random distribution for individual variables, and the 3D scatterplot shows the distribution of full design sets in a space with three of the variables. As can be seen, the distribution is not skewed, which is what was desired.

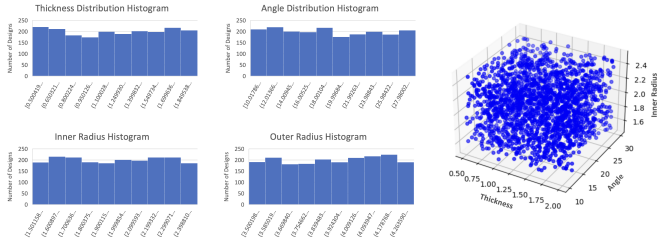


Fig. 9 Randomized Gear Dimension Distributions

The number of random designs to use for ML training in this study was chosen after initial exploratory experimentation. The Mean Squared Error (MSE) of all the ML models of this study dropped significantly by the time 500 designs were used for training, and the MSE was stable by around 800 designs. This was true for all ML models of this study and for both Mass and Displacement computations. Figure 10 shows a sample graph of MSE as a function of ML training data size from testing the Neural Network ML model. Similar results were seen for all ML models. As a result, 2000 designs were chosen for this study as it was highly likely MSE levels would be stable with that training data size.

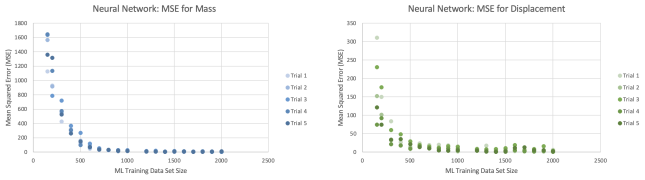


Fig. 10 MSE Values for Mass and Displacement as a Function of ML Training Data Size–Samples from Neural Networks

The six materials used were:

- Plain Carbon Steel
- Steel Alloy
- Aluminum Alloy 1060
- Aluminum Alloy H14
- Copper
- Cast Iron

These are materials with varying physical properties, that are very commonly used in automobile, aircraft, and other industries. Various types of steel are the most commonly used material for components requiring strength[9,60]. Plain carbon steel and the slightly lighter steel alloy were chosen for this reason. Aluminum alloys are frequent substitutes for steel when lower mass is desired[9,60], and so two types of aluminum alloy were chosen for this study. Copper was chosen because it is often the material of choice for gears when corrosion resistance

© The National High School Journal of Science 2025

is important[16 60]. Cast iron was chosen because it is a material frequently used for gears when low cost, good dampening properties, and ease of manufacture are most important[60 61 62].

SolidWorks, an industry leading CAD software package, was used for this work. It was used to create the initial 3D model of a lightweight spur gear with eight triangular cutout openings. The dataset of 2000 random design parameters from Python was imported into SolidWorks. For each of the random design parameters, the mass and displacement of the gear were computed using FEA. Displacement is a measure of how much the gear deformed as a result of the application of force perpendicular to a gear tooth surface. This simulates the force a gear would experience while connected to another gear. For this experiment, 1 N of force was used. The combination of random design values and their FEA computed mass and displacement, served as the starting set of mixed data for surrogate ML model training.

The mixed data was then imported into Python for experimentation. The Python libraries used were numpy, pandas, sklearn, matplotlib, autogluon, keras, and pymoo. The starting data was separated with 80% used for ML model training and 20% used for ML model testing.

Algorithm 1 was tested by first training one AutoGluon ML model with all the mixed training data. One genetic multi-objective optimization (MOO) process with a randomization seed, was run using the one AutoGluon ML model. This generated one Pareto-optimized set of results (process outlined in Figure 6). The results were saved to a csv file. The randomization seed determines a random starting state for the optimization process. In case the starting state influenced the outcomes, the MOO process was run a total of eight times with a different randomization seed each time, leading to eight trials of experimental data for Algorithm 1.

This same approach was used for Algorithms 2 and 3, but using the Keras Neural Network ML model and Random Forest ML model respectively. For each algorithm, a total of eight trials of experimental data, using different randomization seeds, were generated and saved to csv files.

Algorithm 4 was tested by separating the numerical data by material category, into six datasets. Six separate AutoGluon ML models were trained with the separated numerical data. Genetic multi-objective optimization with a randomization seed, was run with the separate AutoGluon ML models leading to six separate sets of Pareto optimized results. The six sets of results were then merged by keeping only the non-dominated results from combined data set (process outlined in Figure 7). These merged results were saved to a csv file. The MOO process was run a total of eight times with a different randomization seed each time, leading to eight trials of experimental data for Algorithm 4.

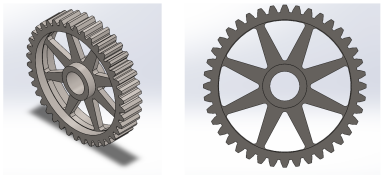
This same approach was used for Algorithms 5 and 6, but using Keras Neural Network ML models and Random Forest ML models respectively. For each algorithm, a total of eight trials of experimental data, using different randomization seeds, were generated and saved to csv files.

To enable accuracy checks, the Pareto-optimal result datasets were imported into SolidWorks. SolidWorks simulations were then run, to compute the mass and displacement values using finite element analysis (FEA) for each of the Pareto-optimal sets of results. Using the ML-MOO generated parameters and the SolidWorks computed values, performance indicators for the Pareto fronts were computed for each of the algorithms. Indicators for accuracy, convergence, and diversity were computed and compared.

3 Results

3.1 Gear Designs Generated by ML-MOO Algorithms

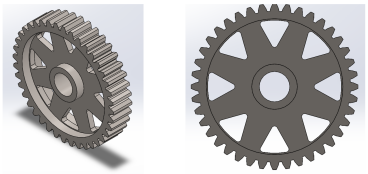
All the algorithms produced similar geometric shapes, as shown in the SolidWorks generated images below. Figure 11 shows a gear with a high displacement and low mass. The web is thin and the size of the cutouts is large, which leads to lower mass and lower structural strength.



Web Thickness	Cutout Inner Radius	Cutout Outer Radius	Cutout Central Angle	Material
0.5003 cm	1.602 cm	4.349 cm	29.76°	Steel Alloy

Fig. 11 High Displacement, Low Mass (20.17×10^{-7} cm, 520.91 grams)

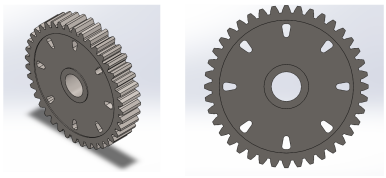
Figure 12 shows a gear that has medium displacement and medium mass. The web is not as thin, and the size of the cutouts is smaller than the cutouts in Figure 11. The cutouts are closer to the edge rather than being close to the center. These design parameters lead to a slightly higher mass and higher structural strength.



Web Thickness	Cutout Inner Radius	Cutout Outer Radius	Cutout Central Angle	Material
0.5017 cm	2.494 cm	4.349 cm	29.94°	Steel Alloy

Fig. 12 Medium Displacement, Medium Mass ($3.946 \times 10 - 7cm, 593.4grams$)

Figure 13 shows a gear that has the lowest displacement and highest mass. The web is as thick as possible, and the size of the cutouts is even smaller than the cutouts in previous gear designs. The cutouts are still close to the edge rather than being close to the center. Overall, these design parameters lead to a much larger mass and greatest structural strength.



Web Thickness	Cutout Inner Radius	Cutout Outer Radius	Cutout Central Angle	Material
1.999 cm	2.498 cm	4.112 cm	10.13°	Steel Alloy

Fig. 13 Low Displacement, High Mass (1.060×10^{-7} cm, 1268 grams)

3.2 Pareto Fronts

The Pareto fronts generated by each of the six algorithms, had the same general shape. Figure 14 shows the Pareto front generated by Algorithm 1 from one of the trials; all eight trials generated similar Pareto fronts. The Pareto front shows that the strongest but also most massive gears used Alloy Steel or Plain Carbon Steel. The lightest but least strong gears used Aluminum Alloy 1060 or Aluminum Alloy H14. Copper gears and Cast Iron gears did not perform well enough to appear on the Pareto front. This graph is similar to ones produced by Algorithms 2 and 3. Sample gear design parameters from three key points (highest mass, middle mass, and lowest mass) from the Algorithm 1 Pareto front are shown in Table 2.

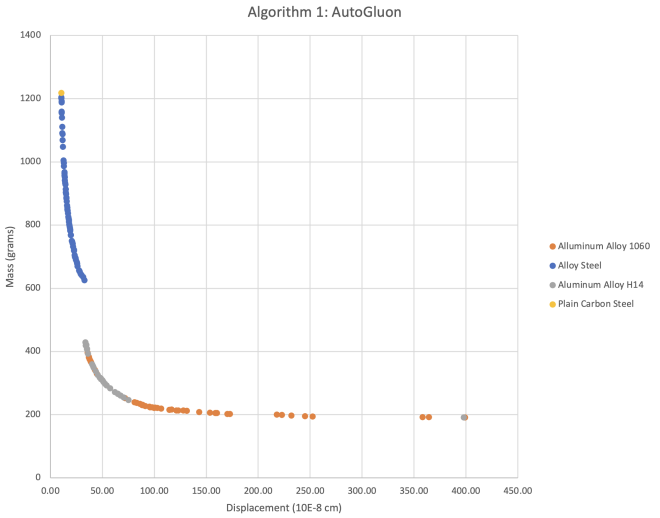


Fig. 14 AutoGluon Pareto Front

Algorithms 4, 5, and 6 used the alternate optimization strategy with multiple ML models and multiple MOO processes, before merging into one combined Pareto front. Figure 15 shows graphs of each of the separate material Pareto fronts from Algorithm 4. Some trends can be seen in this graph. Plain Carbon Steel is good for low displacement gears, but tends to have a high mass. Alloy Steel is similar, but provides the same displacement with slightly lower mass. The two aluminum alloys provide the lowest mass gears but do not provide as much strength as steel. Copper, while having special properties like corrosion resistance that make it a good choice for some situations, is dominated by the other materials in terms of mass and displacement performance. The same is true for Cast Iron.

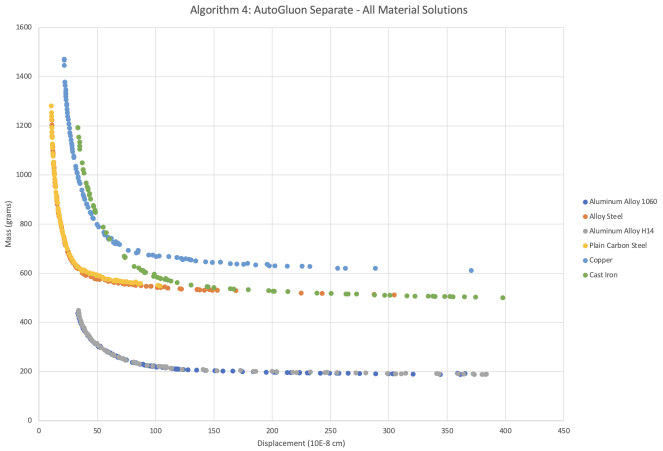


Fig. 15 Pareto Fronts of Separate Materials

Figure 16 shows the final Pareto front generated by Algorithm 4, where all the separate material solutions were combined, and only non-dominated design solutions were kept. None of the Copper and Cast Iron solutions were kept. It has a similar overall shape as the previous Pareto fronts.

3.3 Comparisons

To compare the quality of Pareto fronts, a set of performance indicators must be used together, because no one indicator is without some weaknesses[63]. While there were many options, those appropriate for this study included indicators for accuracy, convergence, and diversity.

Accuracy is a measure of how close the objective values computed by the MOO process using surrogate ML models, are to the values computed by finite element analysis (FEA) run in CAD software, for the same input gear parameter values. The accuracy of each point on the Pareto front, which is a combination of a (a) mass computation and (b) displacement computation, was measured using Mean Square Error (MSE) and the Coefficient of Determination (R^2). The formula for the MSE is:

© The National High School Journal of Science 2025

Table 2 AutoGluon Pareto Front: Sample Gear Design Parameters of Key Points

	Highest Mass Point	Middle Mass Point	Lowest Mass Point
Design Parameters			
Web Thickness	1.982 cm	1.530 cm	0.5096 cm
Cutout Inner Radius	2.231 cm	2.200 cm	1.718 cm
Cutout Outer Radius	4.317 cm	4.320 cm	4.347 cm
Cutout Central Angle	10.02°	16.87°	28.98°
Material	Alloy Steel	Alloy Steel	Aluminum Alloy H14
Performance Outputs			
Mass	1219 grams	625.17 grams	188.76 grams
Displacement	10.66×10^{-8} cm	34.24×10^{-8} cm	385×10^{-8} cm

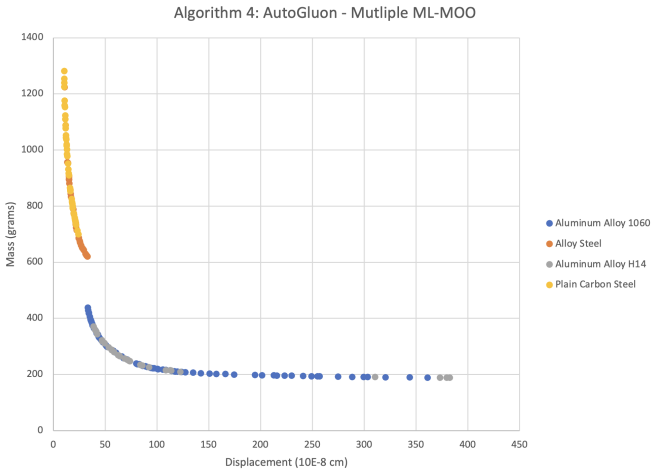


Fig. 16 Multiple AG ML-MOO Pareto Front

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

A lower MSE is better. The formula for the R^2 is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2}$$

R^2 is a measure of how well the ML model accounts for the variance of values. $R^2 = 1$ means the ML model accounts for all the variance and is a perfect fit. $R^2 = 0$ means the ML model doesn't explain any variance, and just predicts the mean of the target. The closer R^2 is to one, the better.

Table 3 shows the MSE and the R^2 across all eight trials. Algorithm 2 (single Neural Network ML model and single MOO) had the lowest mean MSE (lower is better) and the highest mean R^2 (higher is better) when computing the mass. Algorithm 1 (single AutoGluon ML model and single MOO) had the next best MSE and R^2 when computing mass. To get a better sense of scale for MSE, mass was measured in grams, so the mean error is roughly the square root of the MSE. For Algorithms 1 and 2, that would be approximately 7.537 grams and 3.033 grams respectively. The heaviest gears had a mass of approximately 1260 grams while the lightest gears had a mass of approximately 520 grams. The percent error then for Algorithm 1 ranged then from approximately 0.599% to 1.45%. The percent error for Algorithm 2 ranged from approximately 0.241% to 0.583%. Error percentages for both algorithms are very low. For the worst performing Algorithm 6, the percent error ranged from 2.57% to 6.23%.

In Table 4, it can be seen that across all eight trials, Algorithm 2 (single Neural Network ML model and single MOO) produced the best mean MSE and mean R^2 when computing displacement, and Algorithm 5 (multiple Neural Network ML models and multiple MOO) produced the second best. It should be noted, that the MSE and R^2 reported in these tables, are the MSE and R^2 for just those points that are along the thin curve that forms each algorithm's Pareto front. This is because MSE and R^2 are being used as performance indicators for the Pareto front points. These "Pareto front MSE and R^2 values" are not the same as the "initial ML model training MSE and R^2 values." MSE and R^2 performance for any model, is not necessarily uniform across an entire search space.

Convergence is how close the approximated Pareto front is to the actual (or true) Pareto front. There are many ways to measure convergence with Audet et al. listing nine of them[63]. Each indicator has its strengths and weaknesses. One of the most widely cited indicators is Generational Distance[64 65]. It is defined as:

$$\text{GD} = \frac{1}{n}\left(\sum_{i=1}^{n} d_i^2\right)^{1/2} \tag{3}$$

where n is the number of points in the approximate Pareto front, d_i is the Euclidean distance between each point and the nearest member of the true Pareto front. Generational Distance (GD) is basically the average of all the smallest Euclidean distances. Table 5 shows a summary of GD comparison results.

Table 3 MSE and the R^2 for Mass Computations of All Six Algorithms

Algorithm	Property	MSE Mean	MSE St. Dev.	R^2 Mean	R^2 St. Dev.
Algorithm 1	Mass	56.80	16.1	0.9995	0.00015
Algorithm 2	Mass	9.200	2.40	0.9999	0.00040
Algorithm 3	Mass	885.1	110.2	0.9918	0.00081
Algorithm 4	Mass	662.2	45.3	0.9932	0.00042
Algorithm 5	Mass	105.7	14.2	0.9989	0.00015
Algorithm 6	Mass	1048.1	33.9	0.9899	0.00036

Table 4 MSE and the R^2 for Displacement Computations of All Six Algorithms

Algorithm	Property	MSE Mean	MSE St. Dev.	R^2 Mean	R^2 St. Dev.
Algorithm 1	Displacement	307.7	128.1	0.9603	0.00930
Algorithm 2	Displacement	25.4	13.7	0.9976	0.00097
Algorithm 3	Displacement	322.2	252.0	0.7606	0.22376
Algorithm 4	Displacement	827.6	85.4	0.9037	0.00922
Algorithm 5	Displacement	82.7	17.6	0.9898	0.00197
Algorithm 6	Displacement	660.3	106.3	0.8940	0.01247

Algorithm 4 produced the lowest mean GD across eight trials, of 0.870 (best). Algorithm 1 produced the next lowest mean GD of 1.005.

Diversity, also sometimes referred to as distribution and spread, is an indicator of how well an approximated front covers the wide range of values of the actual front[63]. Audet et al. list 25 indicators[63]. However, Zitzler et al. claim that there is no single diversity indicator that allows any conclusions to be drawn about the dominance relationship between two Pareto fronts[65]. There does exist an indicator that captures both convergence and diversity together, the hypervolume (HV) indicator[66,67], that allows conclusions[65]. HV is one of the most widely cited indicators[63] in research on Pareto fronts. Figure 17 shows how the hypervolume is computed for a set of points based on a chosen reference point[68]. Figure 18 shows how the choice of reference point can affect the contribution of the end points to the total hypervolume[68]. To aim for approximately equal contribution by all points to the hypervolume, Ishibuchi et al. recommend using reference point values (r,r), using the formula $r = r + 1/(n-1)$, where n is the total number of Pareto front values[68]. This is in the ideal situation where f_1 and f_2 have the same endpoints and scales. In other situations, each axis can be scaled separately with $r_1 = r_1(1 + 1/(n-1))$ and $r_2 = r_2(1 + 1/(n-1))$. This practice was followed in this experiment, because one axis is mass in grams while the other axis is displacement in 10^{-7} cm. It is impossible for both axes to have identical end points and scale.

Looking at hypervolumes (HV), Table 6 shows a summary of HV comparison results. Algorithm 4 produced the highest mean HV of 377,758 (best). Algorithm 1 produced the next highest mean HV of 377,075.

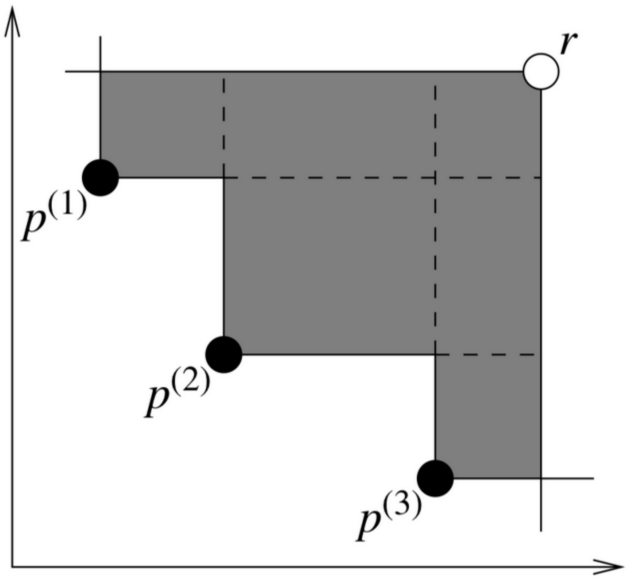


Fig. 17 Hypervolume Computation[68]

The hypervolume measure was also used to determine how long the MOO processes should be run for the main experiment. To ensure a fair comparison between algorithms, an identical number of ML model evaluations (or model "calls") were used for all MOO processes. This number needed to be high enough to ensure that stable Pareto fronts had been achieved. This fair comparison point was determined by comparing hypervolume performance across Algorithms 1, 2, and 3. Figure 19 shows the hypervolume performance of Algorithms 1, 2, and 3, as a

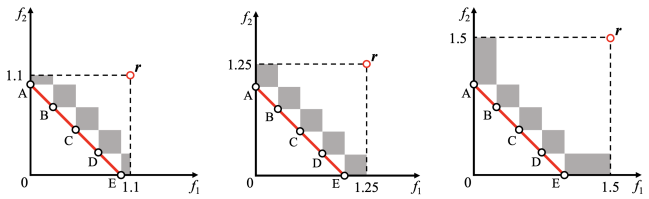
© The National High School Journal of Science 2025

Table 5 Generational Distances (GD) Means of All Six Algorithms

Algorithm	GD Mean of 8 Trials	GD St. Dev.
Algorithm 1: Single ML model single MOO with AutoGluon	1.005	0.299
Algorithm 2: Single ML model single MOO with Keras Neural Networks	1.973	0.276
Algorithm 3: Single ML model single MOO with Decision Trees (Random Forest)	2.218	0.32
Algorithm 4: Multiple ML multiple MOO with AutoGluon	0.870	0.33
Algorithm 5: Multiple ML multiple MOO with Keras Neural Networks	2.514	0.692
Algorithm 6: Multiple ML multiple MOO with Decision Trees (Random Forest)	3.453	0.288

Table 6 Hypervolumes (HV) Means of All Six Algorithms with Reference Point (402, 1210)

Algorithm	HV Mean of 8 Trials	HV St. Dev.
Algorithm 1: Single ML model single MOO with AutoGluon	377,075	100.55
Algorithm 2: Single ML model single MOO with Keras Neural Networks	376,779	120.64
Algorithm 3: Single ML model single MOO with Decision Trees (Random Forest)	373,314	975.75
Algorithm 4: Multiple ML multiple MOO with AutoGluon	377,758	23.65
Algorithm 5: Multiple ML multiple MOO with Keras Neural Networks	375,895	47.29
Algorithm 6: Multiple ML multiple MOO with Decision Trees (Random Forest)	374,316	61.24



(a) $r = 1.1$. (b) $r = 1.25$. (c) $r = 1.5$.

Fig. 18 Hypervolume Contribution Variations Depending on Choice of Reference Point[68]

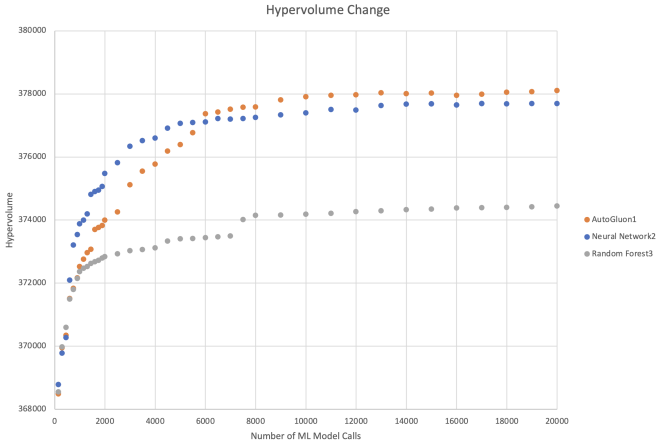


Fig. 19 Hypervolume by Number of ML Model Calls

function of the number of ML model evaluations. As expected, hypervolumes of the generated Pareto fronts generally increased as more ML model evaluations were performed. Algorithm 2 showed a more rapid increase in performance initially, but by 6,000 evaluations, it was surpassed by Algorithm 1. At around 10,000 evaluations, the hypervolume performance of all three algorithms stabilizes. The change in hypervolume after that, is never more than 0.039% for every increase in 1000 ML model calls. Figure 20 shows the same hypervolume performance changes, but for all eight trials with different randomization seeds. Below 10,000 evaluations, the ranking and HV values varied depending on the trial. But in all eight cases, the performance and ranking of all three algorithms stabilizes at around 10,000 evaluations. For this reason, 10,000 evaluations was chosen as a fair comparison point for all the Pareto front performance indicators computed for the six algorithms in this experiment.

A related helpful measure is a hypervolume's contribution rate (CR) indicator[67]. The CR indicator provides a score on a scale of zero to one, where the higher value is best. This indicator helps compare a group of competing Pareto fronts by providing a number that shows the fraction of values that a particular Pareto front contributes to the best combined Pareto front of the entire group[67]. If $PF_1, PF_2, ...PF_n$ refer to competing Pareto fronts, then PF_s is a single set of non-dominated values of a combined Pareto front. PF_s serves as a "surrogate" for the true Pareto front. $PF'_1, PF'_2, ...PF'_n$ are then computed where $PF'_i = PF_i \cap PF_s$. The contribution rate, CR, can then be defined as:

$$CR(PF_i, PF_s) = \frac{\text{hypervolume}(PF'_i)}{\text{hypervolume}(PF_s)} \quad (4)$$

A "surrogate" true Pareto front can be constructed by combining only non-dominated designs from all six algorithms[67].

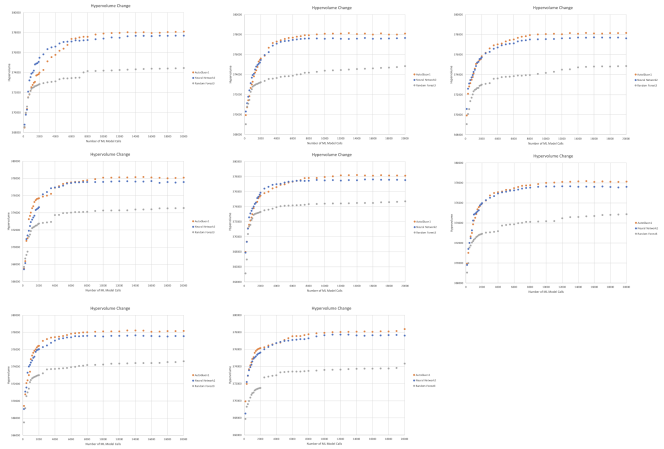


Fig. 20 Hypervolume by Number of ML Model Calls

Actual true Pareto fronts are rarely known, because determining them would require prohibitively expensive and time-consuming physical testing of thousands of physical design variations. As a result, in many situations, the closest approximations available must be used to serve as surrogate true Pareto fronts[67]. FEA generated values are known to be highly accurate (ranging in many situations between 90% and 98%)[18 19] and so using FEA values to construct a surrogate true Pareto front is acceptable. Figure 21 shows the surrogate true Pareto front for this study, with each point color coded to the algorithm that contributed that solution. Algorithm 4 contributed the most points that spanned the entire Pareto front. Other algorithms, like Algorithm 5, had contributions concentrated in a certain section of the Pareto front.

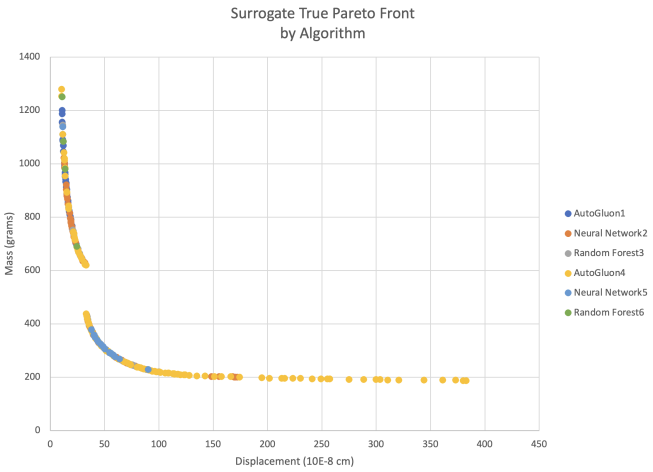


Fig. 21 Combined "Surrogate" Pareto Front by Algorithm

Table 7 shows a summary of the number of solutions contributed by each algorithm to the surrogate Pareto front and the Contribution Rate of each algorithm. Algorithm 4 contributed the most solutions with a mean over eight trials of 130.3 non-dominated points, while Algorithm 1 contributed the second most with a mean of 71.9. Algorithm 1 contributed more points than the more manual-labor-intensive Algorithms 5 and 6.

Finally, Table 8 shows the mean runtimes over all 8 trials of the MOO process and total number of ML evaluations from each of the six algorithms. Runtimes are in seconds, with Algorithm 4 having the longest runtime of 185.3 seconds (3.09 minutes).

4 Discussion

When considering all the performance indicators together, Algorithm 4 (multiple AutoGluon) performed the best. It had the best Hypervolume (HV), Generational Distance (GD), and Contribution Rate (CR). The performance values for Mean Square Error (MSE) and Coefficient of Determination (R^2) for Algorithm 4 were in the bottom third. However, when considering the group of indicators together, Algorithm 4 provided the best performance overall. The indicators show that Algorithm 4 generated a Pareto front that was closest to the true Pareto front.

The MSE of Algorithm 4 does not cause it to rank below Algorithm 2 (best MSE), because even with less accurate mass and displacement predictions, Algorithm 4's ML predicted values and FEA computed values, are both closer to the true Pareto front that Algorithm 2's ML predicated values and FEA computed values. Algorithm 2's Pareto front ML predicted values and FEA computations may be closer to each other, but they are both further from the true Pareto front than Algorithm 4's Pareto front points. Also, it should be remembered once again that these MSE values are for just the points along the thin curve that forms each algorithm's Pareto front. They are not the MSE values for the ML models for the larger search space.

These overall results were true across all eight trials of the experiment. While there was some variability in performance indicator values between trials (as indicated by the reported standard deviation), the overall results were still consistent. For all of the trials, the relative order of outcomes between algorithms, for every performance indicator, remained exactly the same.

Algorithm 1 (single AutoGluon) performed second best. It had the second best performance values for Hypervolume (HV), Generational Distance (GD), and Contribution Rate (CR), and top third performance values for Mean Square Error (MSE) and Coefficient of Determination (R^2). It actually performed better than Algorithm 4 (multiple AutoGluon) for MSE and R^2.

It was surprising that Algorithm 1 performed second best, even better than Algorithms 5 and 6. Every algorithm used 10,000 ML model evaluations for each MOO process. So with the six separate ML models (see Figure 7), Algorithms of 5 and 6 each used 60,000 total ML model evaluations to generate their Pareto front. By comparison, Algorithm 1 used just one ML model and produced better results with just 10,000 total ML model evaluations. One might assume that algorithms that

© The National High School Journal of Science 2025

Table 7 Contributions to the Surrogate Combined "True" Pareto Front Summary

Algorithm	Number of non-dominated solutions		Contribution Rate	
	Mean	St. Dev.	Mean	St. Dev.
Algorithm 1	71.9	6.33	0.9918	0.003
Algorithm 2	27.0	8.64	0.967	0.019
Algorithm 3	21.0	3.85	0.9578	0.005
Algorithm 4	130.3	11.41	0.9986	0.001
Algorithm 5	22.0	4.07	0.9509	0.011
Algorithm 6	4.6	2.77	0.4934	0.357

Table 8 MOO Runtime Means

	Total Number of Evaluations	Runtime Mean (seconds)	Runtime St. Dev.
Algorithm 1	10,000	44.69	0.464
Algorithm 2	10,000	5.948	0.141
Algorithm 3	10,000	2.088	0.064
Algorithm 4	60,000	185.3	0.207
Algorithm 5	60,000	43.48	0.337
Algorithm 6	60,000	10.15	0.207

used six times as many evaluations would perform better. But that was not the case here. As far as why, it could be that since an individual ML model has various strengths and weaknesses, the stacked-ensemble AutoGluon model was able to very effectively combine various ML model outputs to produce an overall better result than Algorithms 5 and 6 – even with 83% fewer computations.

Another unexpected result was that Algorithm 2 (single Neural Network) did better than Algorithm 5 (multiple Neural Network), and that Algorithm 3 (single Random Forest) did better than Algorithm 6 (multiple Random Forest) for GD and CR. Algorithm 2 used the same Neural Network ML model as Algorithm 5, but used 83% fewer computations. Algorithm 3 used the same Random Forest ML model as Algorithm 6 but also used 83% fewer computations. One would expect the algorithm with more total computations to have better convergence and contribute more than an algorithm with fewer computations. However, it could instead be the case that the relationships between the numerical and categorical data in the mixed dataset, was captured by the ML models of Algorithms 2 and 3, leading to better overall performance than their Algorithm 5 and 6 counterparts.

In summary, the AutoGluon ML models were the easiest to use, not requiring any time to be spent on manual adjusting of hyperparameters to improve performance. While all six algorithms produced good results with mixed data, as shown by the graphs and indicator values, the algorithms with the easiest to use AutoML models, produced the best results. An implication of this is that engineers can engage in data-driven design, without needing to have advanced technical knowledge of ML model hyperparameter tuning. This makes data-driven design more accessible to a wider audience of designers.

This experiment also revealed some information about lightweight gear materials and geometry. A set of commonly used gear materials was chosen for the experiment. They were Plain Carbon Steel, Steel Alloy, Aluminum Alloy 1060, Aluminum Alloy H14, Copper, and Cast Iron. It was surprising to see copper and cast iron not performing well enough to appear in the surrogate true Pareto front. In terms of just strength and mass performance for gears, the steel options dominated in the higher mass section of the Pareto front and aluminum options dominated the lower mass section of the Pareto front (see Figure 16). However, as shown in Figure 15, copper and cast iron gears can also be optimized for desired performance characteristics through geometric choices. Though not as good as steel and aluminum, if their other positive properties (like corrosion resistance) are desired, engineers can use a Pareto front to guide their design work. For gear geometry, as expected, the lightest gears had large cutouts and thin webs. It was interesting to see, and reasonable, that as the gears became stronger, the cutouts tended to cluster in positions on the outside of the gears, rather than towards the middle of the gears.

Limitations of this study include possible experimental error from the choice of randomization seeds for the MOO process. Eight MOO randomization seed values were used for this experiment for all algorithms. An assumption for this experiment was that any seed would produce results that are typical and similar to the results produced from choosing other seeds. While this proved true for these eight seeds, this assumption could be incorrect when a much larger number of seeds is checked. The choice

of seeds could lead to different results for the other performance indicators. Future work to check this possible error would involve running the experiment using a much larger number of seeds to see if there are any effects on the outcomes.

The results of this experiment can help those in industry who are trying to design more efficient lightweight parts and machines. The results show that it is possible to use an automated ML-MOO approach to get high quality mechanical gear designs from mixed data. These designs can be better than the designs generated by more human-attention-intensive ML-MOO algorithms. All of the algorithms were low cost and much faster than physically manufacturing and testing gear designs. All of the algorithms were faster than CAD simulations using FEA. Based on timing SolidWorks FEA performance for smaller batches and scaling up, 60,000 model simulations would have taken approximately 170 hours, or 7.08 days. By comparison, as shown in Table 8, all the algorithms in this experiment completed 60,000 ML model evaluations in just a few minutes. This significantly faster performance may encourage more engineers to use data-driven design methods to improve their own designs. This will help our overall effort to make better use of Earth's limited natural resources.

In the future, investigating these methods for other mechanical parts with different geometries and materials would be valuable. These additional investigations could follow the same general procedure, with additional parameters related to the new mechanical part contexts added. New CAD models would need to be created, but the ML models and MOO processes would function the same way. Investigating additional AutoML models would also be useful. AutoGluon is just one example of a stacked-ensemble model, and it would be interesting to compare its performance with that of others. Finally, a more detailed study investigating how the properties of ML models influence the MOO process could be of value. As noted above, ML model accuracies are not uniform across the entire search space, and more a precise understanding the necessary patterns of accuracies for tasks like the MOO process could be useful.

5 Acknowledgments

I would like to thank Mr. Lyle Regenwetter, Ph.D. candidate in Mechanical Engineering at MIT, for guiding me through the process of creating CAD designs, training ML models, and using multi-objective optimization. I would also like to thank Ms. Gupreet Juneja from Adlai E. Stevenson High School, for helping me with the research and writing process for this work.

Appendix

Below are the hyperparameter settings for the three ML models used in this study.

AutoGluon

- Preset of 'good_quality" used.
- Metric of 'mean_absolute_error'
- Time limit of 1600

Keras Neural Networks

- Feedforward neural network using Kera's Sequential API with an input layer, two hidden layers, and an output layer. Separate ML models trained for mass computations and displacement computations.
- Input Layer for gear input parameter data.
- Hidden Layer 1
  - 64 neurons
  - ReLU activation
- Hidden Layer 2
  - 32 neurons
  - ReLU activation
- Output Layer
  - Single neuron output for regression
- Optimizer: Adam with a learning rate of 0.001
- Loss function: 'mean_squared_error'

Random Forest

- Used RandomForestRegressor
- 30 decision trees
- Unlimited maximum depth
- Minimum samples to split an internal node of 2
- Minimum samples at a leaf node of 1
- Fixed random state (for reproducibility)

References

1 J. Kagathara, T. Lubben and M. Steinbacher, *Effect of design modification on the distortion behavior of a complex counter gear*.

2 G. Xu and N. Dai, *Michell truss design for lightweight gear bodies*.

3 L. Regenwetter, C. Weaver and F. Ahmed, *FRAMED: An AutoML approach for structural performance prediction of bicycle frames*.

© The National High School Journal of Science 2025

4 L. Liu, Z. Li, H. Kang, Y. Xiao, L. Sun, Z. Zhao and Y. Ma, *Review of surrogate model assisted multi-objective design optimization of electrical machines: New opportunities and challenges*.

5 S. Oh, Y. Jung, S. Kim, I. Lee and N. Kang, *Deep generative design: Integration of topology optimization and generative models*.

6 M. Cheng, X. Zhao, M. Dhimish, W. Qui and S. Niu, *A review of data-driven surrogate models for design optimization of electric motors*.

7 A. Candela, G. Sandrini, M. Gadola, D. Chindamo and P. Magri, *Lightweighting in the automotive industry as a measure for energy efficiency: Review of the main materials and methods*.

8 F. Cosco, R. Adduci, L. Muzzi, A. Rezayat and D. Mundo, *Multiobjective design optimization of lightweight gears*.

9 F. Czerwinski, *Current trends in automotive lightweighting strategies and materials*.

10 J. Wang, Y. Li, G. Hu and M. Yang, *Lightweight research in engineering: A review*.

11 D. Politis, N. Politis and J. Lin, *Review of recent developments in manufacturing lightweight multi-metal gears*.

12 *New Equipment Digest. NASA's revolutionary hybrid gear lightens your load*, https://www.newequipment.com/research-and-development/article/22058083/nasas-revolutionary-hybrid-gear-lightens-your-load.

13 R. Ramadani, A. Belsak, M. Kegl, J. Predan and S. Pehan, *Topology optimization based design of lightweight and low vibration gear bodies*.

14 H. Hobby, *64P Featherweight Aluminum Pinion Gear*, https://www.horizonhobby.com/product/64p-featherweight-aluminum-pinion-gear-70t/TEP6470.html.

15 TaiwanGun, *Ultra Lightweight Titanium Gear Set*, https://www.taiwangun.com/gears/ultra-lightweight-titanium-gear-set-16-1-bd.

16

17 B. Hund and A. Woldeyohannes, *Future prospects of computer-aided design (CAD) - A review from the perspective of artificial intelligence (AI), extended reality, and 3D printing*.

18 A. Palazotto, E. Herup and L. Gummadi, *Finite element analysis of low-velocity impact on composite sandwich plates*.

19 Y. Gur and G. Cen, *Comparison of finite element analysis results with strain gauge measurements of a front axle housing*.

20 U. Idrees, S. Ahmad, M. I.Shah, R. Shehzad, M. Amjad and S. Koloor, *Finite element analysis of car frame frontal crash using lightweight materials*.

21 P. Badarinath, M. Chlerichetti and F. Kakhki, *A machine learning approach as a surrogate for finite element analysis: status of research and application to one dimensional systems*.

22 Y. Jin, H. Wang, T. Chugh, D. Guo and K. Miettinen, *Data-driven evolutionary optimization: An overview and case studies*.

23 U. Urbas, D. Zorko and N. Vukasinovic, *Machine learning based nominal root stress calculation model for gears with a progressive curved path of contact*.

24 C. Collier and S. Jones, *Unified analysis of aerospace structures through implementation of rapid tools into a stress framework*.

25 P. Badarinath, M. Chierichetti and F. Kakhki, *A machine learning approach as a surrogate for a finite element analysis: status of research and application to one dimensional systems*.

26 S. Bandyopadhyay, R. Chakraborty and U. Maulik, *Priority based dominance: A new measure in multiobjective optimization*.

27 P. Stabile, F. Ballo, M. Gobbi and G. Previati, *Multi-objective structural optimization of vehicle wheels: a method for preliminary design*.

28 W. Chen, K. Chui and M. Fuge, *Aerodynamic design optimization and shape exploration using generative adversarial networks*.

29 G. Bonori, M. Barbieri and F. Pellicano, *Optimum profile modifications of spur gears by means of genetic algorithms*.

30 D. Miler, D. Zezilj, A. Loncar and K. Vuckovic, *Multi-objective spur gear pair optimization focused on volume and efficiency*.

31 R. Renz and A. Albers, *Multi-material topology optimization on separate tetrahedral meshes with explicit design resolution by means of remeshing*.

32 M. Elsiedy, H. Hegazi, A. El-Kassas and A. Zayed, *Multi-objective design optimization of polymer spur gears using a hybrid approach*.

33 N. Sokolov, *Finite element analysis of stress distribution in optimized gear systems*.

34 K. Pillai, A. Ray, S. Kaul and N. Babu, *Design optimization of spur gear using genetic algorithm*.

35 W. Xin, Y. Zhang, Y. Fu, W. Yang and H. Zheng, *A multi-objective optimization design approach of large mining planetary gear reducer*.

36 J. Manson, T. Chamberlain and R. Bourne, *MVMOO: Mixed variable multi-objective optimization*.

37 N. Bartoli, T. Lefebvre, R. Lafage, P. Saves, Y. Diouane, J. Morlier, J. Bussemaker, G. Donelli, J. Mello, M. Mandorino and P. Vecchia, *Multi-objective Bayesian optimization with mixed-categorical design variables for expensive-to-evaluate aeronautical applications*, arXiv:2504.09930.

38 H. Li, Z. Liu and P. Zhu, *An improved multi-objective optimization algorithm with mixed variables for automobile engine hood lightweight design*.

39 P. Saves, N. Bartoli, Y. Diouane, T. Lefebvre, J. Morlier, C. David, E. Van and S. Defoort, *Multidisciplinary design optimization with mixed categorical variables for aircraft design*.

40 O. Kershaw, A. Clayton, J. Manson, A. Barthelme, J. Pavey, P. Peach, J. Mustakis, R. Howard, T. Chamberlain, N. Warren and R. Bourne, *Machine learning directed multi-objective optimization of mixed variable chemical systems*.

41 S. Aryal and J. Wells, *Ensemble of local decision trees for anomaly detection in mixed data*.

42 A. Rosebrock, *Keras: Multiple inputs and mixed data*, https://pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/.

43 B. Chicho and A. Sallow, *A comprehensive survey of deep learning models based on Keras framework*.

44 Q. Yao, M. Wang, H. Escalante, I. Guyon, Y. Hu, Y. Li, W. Tu, Q. Yang and Y. Yu, *Taking human out of learning applications: a survey on automated machine learning*.

45 O. Daanouni, B. Cherradi and A. Tmiri, *Predicting diabetes diseases using mixed data and supervised machine learning algorithms*.

46 O. Apolo-Apolo, M. Perez-Ruiz, J. Martinez-Guanter and G. Egea, *A mixed data-based deep neural network to estimate leaf area index in wheat breeding trials*.

47 M. Ahsan, T. Alam, T. Trafalis and P. Huebner, *Deep MLP-CNN model using mixed-data to distinguish between COVID-19 and non-COVID-19 patients*.

48 K. Rojewska, *What are neural networks and what are their applications?*, https://www.qtravel.ai/blog/what-are-neural-networks-and-what-are-their-applications/.

49 M. Zoller and M. Huber, *Benchmark and survey of automated machine learning frameworks*.

50 M. Feurer and F. Hutter, *Hyperparameter optimization*.

51 X. He, K. Zhao and X. Chu, *AutoML: A survey of the state-of-the-art*.

52 N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li and A. Smola, *AutoGluon-Tabular: robust and accurate AutoML for structured data*.

53 H. Mendoza, A. Klein, M. Feurer, J. Springenberg, M. Urban, M. Burkart, M. Dippel, M. Lindauer and F. Hutter, *Towards automatically-tuned deep neural networks*.

54 O. Owoyele, P. Pal, A. Torreira, D. Probst, M. Shaxted, M. Wilde and P. Senecal, *Application of an automated machine learning-genetic algorithm (AutoML-GA) coupled with computational fluid dynamics simulations for rapid engine design optimization*, arXiv:2101.02653.

55 R. Subramanya, *A data-driven design pipeline for the structural optimization of quadcopters*.

56 C. Daniel, *AutoGluon-enabled machine learning models for predicting recycled aggregate concrete's compressive strength*.

57 N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li and A. Smola, *AutoGluon-Tabular: Robust and accurate AutoML for structured data*, arXiv:2003.06505.

58 J. Wang, Q. Xue, C. Zhang, K. Wong and Z. Lui, *Explainable coronary artery disease prediction model based on AutoGluon from AutoML framework*.

59 MiMOTION, *Gears*, https://www.motion.com/products/Mechanical%20Power%20Transmission/Drive%20Components/Gears.

60 I. Dupree, *Gear Material Selection for Optimal Performance*, https://techmestuff.com/gear-material-performance-optimize/.

61 S. Lee, *Gear Materials 101: A beginner's guide*, https://www.numberanalytics.com/blog/gear-materials-101.

62 P. Jana and P. Dan, *Optimization treatment of material selection*.

63 C. Audet, J. Bigeon, D. Cartier, S. Digabel and L. Salomon, *Performance indicators in multiobjective optimization*.

64 D. Veldhuizen and G. Lamont, *On measuring multiobjective evolutionary algorithm performance*.

65 E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca and V. Fonseca, *Performance assessment of multiobjective optimizers: an analysis and review*.

66 E. Zitzler and L. Thiele, *Multiobjective optimization using evolutionary algorithms - A comparative study*.

67 Y. Cao, B. Smucker and T. Robinson, *On using the hypervolume indicator to compare Pareto fronts: Applications to multi-criteria optimal experimental design*.

68 H. Ishibuchi, R. Imada, Y. Setoguchi and Y. Nojima, *Reference point specification in hypervolume calculation for fair comparison and efficient search*.

© The National High School Journal of Science 2025