

Review

Engineering Applications of Artificial Intelligence in Mechanical Design and Optimization

Jozef Jenis ^{1,*}, Jozef Ondriga ¹, Slavomir Hrcek ¹, Frantisek Brumercik ¹, Matus Cuchor ¹ and Erik Sadovsky ²

¹ Department of Design and Mechanical Elements, Faculty of Mechanical Engineering, University of Žilina, 010 26 Žilina, Slovakia; jozef.ondriga@fstroj.uniza.sk (J.O.); slavomir.hrcek@fstroj.uniza.sk (S.H.); frantisek.brumercik@fstroj.uniza.sk (F.B.); matus.cuchor@fstroj.uniza.sk (M.C.)

² Department of Multimedia and Information-Communication Technologies, Faculty of Electrical Engineering and Information Technology, University of Žilina, 010 26 Žilina, Slovakia; erik.sadovsky@feit.uniza.sk

* Correspondence: jozef.jenis@fstroj.uniza.sk

Abstract: This study offers a complete analysis of the use of deep learning or machine learning, as well as precise recommendations on how these methods could be used in the creation of machine components and nodes. The examples in this thesis are intended to identify areas in mechanical design and optimization where this technique could be widely applied in the future, benefiting society and advancing the current state of modern mechanical engineering. The review begins with a discussion on the workings of artificial intelligence, machine learning, and deep learning. Different techniques, classifications, and even comparisons of each method are described in detail. The most common programming languages, frameworks, and software used in mechanical engineering for this problem are gradually introduced. Input data formats and the most common datasets that are suitable for the field of machine learning in mechanical design and optimization are also discussed. The second half of the review describes the current use of machine learning in several areas of mechanical design and optimization, using specific examples that have been investigated by researchers from around the world. Further research directions on the use of machine learning and neural networks in the fields of mechanical design and optimization are discussed.



Citation: Jenis, J.; Ondriga, J.; Hrcek, S.; Brumercik, F.; Cuchor, M.; Sadovsky, E. Engineering Applications of Artificial Intelligence in Mechanical Design and Optimization. *Machines* **2023**, *11*, 577. <https://doi.org/10.3390/machines11060577>

Academic Editor: Ibrahim N Tansel

Received: 26 April 2023

Revised: 11 May 2023

Accepted: 16 May 2023

Published: 23 May 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Mechanical design is one of the most essential areas of mechanical engineering. The building or design of different machines and equipment may be described as the process by which resources or energy are transformed into usable mechanical shapes or processes to produce useful output from machines, in accordance with human demands. The design of a given machine might result in the production of an entirely new machine or its enhancement (optimization). All of these ideas and optimization techniques are incorporated in a multitude of software solutions currently available on the market. The most well-known are CAx systems, which can replicate the different phases of the life cycles of individual components as well as full machines and equipment in the virtual environment.

However, it is the machine designer who transforms an idea into reality. It is their responsibility to ensure that the device works properly, has the right features, and is as cheap as possible. At present, however, technical problems are very sophisticated and complex, especially when it comes to obtaining a suitable solution to the task. This is one of the reasons why designers often make mistakes, i.e., they are pressed by time, small budgets, or insufficient knowledge of the issue. This is one of the reasons why a modern designer should not only rely on outdated catalogs, machine tables, and outdated procedures, but

should strive for progress by adapting to modern methods that are available today thanks to the rapid evolution of information technology.

However, to prevent errors and inconsistencies in the design process itself, automation, artificial intelligence, and machine learning can be used in practice, which significantly reduce development times, save costs, and, most importantly, minimize the error rate throughout the design process; moreover, information can be processed by a specific algorithm. Thus, information and knowledge no longer need to be collected and processed by a human (the designer).

This study, therefore, presents a thorough overview of the use of deep learning or machine learning and specific suggestions on how these techniques can be used in the design of machine components and nodes. Machine learning is a fundamental component of artificial intelligence. The examples in this thesis aim to look for opportunities in the field of mechanical design and optimization, where this technique could be commonly applied in the future, ultimately benefiting society and the modern world of mechanical engineering.

Machine learning, or deep learning, is already being used today in various areas of everyday life. There is also continuous research on how to improve the outputs of such applications and technologies. As an example, research was conducted on cluster-based multidimensional approaches for detecting attacks on connected vehicles [1], where the researchers presented two algorithms that implemented an anomaly detection system based on neural network data. We can also cite a research paper conducted in the field of coronavirus diseases [2], where researchers used deep learning to accelerate the development of a cure for COVID-19 by developing new algorithms to detect different genomes of SARS-CoV-2. Of course, we cannot overlook the area of diagnostics, where machine learning and artificial intelligence are often applied. A good example is the 2020 general overview titled “Applications of machine learning to machine fault diagnosis: A review and roadmap” [3], where researchers proposed a comprehensive diagnostic procedure for machine health detection.

2. Artificial Intelligence and Machine Learning

The field of artificial intelligence (AI) (see Figure 1) is gaining a lot of attention due to its ability to effectively analyze and act on the vast amount of data collected. It has been a subject of research since the 1950s. The rapid increase in interest is mainly attributed to advances made in the machine learning (ML) sub-area, as well as supporting factors, such as data storage and computing power.

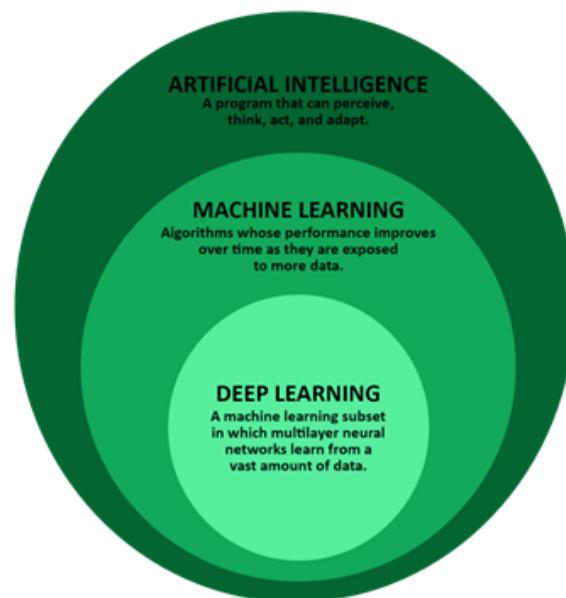


Figure 1. Artificial intelligence, machine learning, and deep learning.

The interest in this technology is great and various subjects present numbers that inform about the high potential of artificial intelligence. The systems of this technology are becoming powerful tools that can be used and can lead to faster, more rational decisions, as well as more efficient work. Therefore, with the further spread of artificial intelligence and machine learning at the organizational and societal levels, significant benefits are likely to be achieved.

2.1. Artificial Intelligence

Artificial intelligence (see Figure 1) permeates our environment. It involves a range of sub-fields, ranging from the abstract to the tangible, including self-driving vehicles, chess games, theorem provers, music, art, etc. AI is one of the most exciting and diverse disciplines of computer science, with a vast future reach. Artificial intelligence has the tendency to make machines behave similar to humans [4].

Artificial intelligence (AI) is an area of computer science that may be used to develop intelligent computers that can act, think, and make choices similar to humans. Artificial intelligence is shown when a machine has human-like abilities, such as learning, thinking, and problem solving [4].

2.1.1. Goals of Artificial Intelligence

- Solving knowledge-intensive tasks;
- Making the connection between perceptions and actions;
- Developing machines that can perform tasks that require human intelligence;
- Creating systems that can exhibit intelligent behavior, learn new things on their own, and demonstrate, explain, and advise their users [4].

2.1.2. Benefits of Artificial Intelligence

The benefits of AI are vast; these are the main ones:

- Reducing the human error rate—in artificial intelligence, decisions are made from pre-collected information using a set of algorithms. As a result, errors are reduced to a minimum and the probability of achieving accuracy with a higher degree of accuracy increases rapidly.
- Risk transfer from people to AI—this is one of the greatest advantages of artificial intelligence. Many high-risk occupations can be overcome via the development of AI, which minimizes the risk of death or injury to humans. For example, a robot with AI can go to Mars, defuse a bomb, explore the deepest parts of the oceans, mine coal and oil, and be used effectively in any natural disasters.
- Continuous operation—the average person works 4–6 h a day without breaks. However, with artificial intelligence, we can ensure that machines work 24 h a day, 7 days a week, without any breaks; unlike humans, they do not even get bored.
- Automation of repetitive tasks—with the help of artificial intelligence, everyday tasks can be productively automated, even “boring” tasks for people can be removed and released so that they become more creative.
- Digital use—there are highly developed enterprises that engage with people through digital assistants, effectively conserving human resources. Digital assistants are also used on many websites, allowing users to talk with them about what they are looking for or need. Some chatbots are designed in a way that makes it difficult to discern whether the user is interacting with a chatbot or a human being.
- Faster decision-making—by using AI with other technologies, machines can make decisions and execute actions faster than humans. When making a decision, one will analyze many factors (both emotionally and factually), but a machine powered by artificial intelligence works according to how it is programmed and delivers results faster [5].

2.1.3. Disadvantages of Artificial Intelligence

Due to the fact that every positive trait has a negative counterpart, artificial intelligence also has significant drawbacks. Here are several examples:

- High creation costs—AI is upgraded daily; hardware and software must be up to date to satisfy the most recent needs. Costly repairs and upkeep are required for machines and systems. Moreover, in general, the use of such technology is prohibitively expensive due to the complexity of its components and systems.
- Creating human laziness—artificial intelligence encourages laziness since its applications automate the majority of labor. People have a tendency to rely on these innovations, which might be problematic for future generations.
- Rising unemployment—as artificial intelligence is replacing the majority of repetitive chores and other forms of employment, human interaction is dwindling, which poses a significant concern for labor standards. Each firm aims to replace low-skilled workers with AI robots that are capable of performing the same tasks more efficiently.
- Absence of emotions—when it comes to job efficiency, robots are superior to humans, but they cannot replace the human relationships that comprise a team. Machines are incapable of forming relationships with people, which is a crucial skill for team management.
- Lack of thinking—the only jobs that machines and systems are capable of doing are those for which they were created or programmed to do. Anything beyond their designated functions may result in crashes or generate irrelevant outputs, which can present significant hurdles [5].

2.2. Machine Learning

Machine learning (ML) (see Figure 1) is a field within artificial intelligence (AI) and computer science that focuses on utilizing data and algorithms to imitate how humans learn and steadily improve accuracy. Data science is an expanding discipline, and machine learning is a crucial component of it. In data mining initiatives, classification and prediction algorithms are trained using statistical approaches, providing crucial insights. This information then facilitates decision-making inside applications and organizations, ideally affecting key growth metrics. With the increasing volume of big data, there will be a greater need for data scientists who can assist in identifying the most crucial business issues and providing answers [6].

2.2.1. The Way Machine Learning Works

The machine learning algorithm-training system consists of three parts:

- Decision-making process: Machine learning techniques are often used for prediction or categorization. Based on particular input data, which may or may not be tagged, the algorithm estimates the data pattern.
- Error function: The error function is used to assess the model's prediction. If examples are available, the error function may conduct a comparison to evaluate the model's correctness.
- Model optimization process: If the model can be better suited to the training set's data points, then the weights are modified to decrease the distance between the known example and the model prediction. The algorithm will continue this assessment and optimization process, updating the weights until a predetermined level of accuracy is attained [6].

2.2.2. Types of Machine Learning

Machine learning is divided into the following types of learning, according to the type of task it solves:

- Supervised learning—this is defined by the use of tagged datasets (see Figure 2) to train algorithms to accurately classify data or predict results. As input data are entered

into the model, the weights are adjusted until the model fits properly. This is done as part of the cross-validation process to ensure that the model avoids over- or under-adaptation. Supervised learning helps organizations address a variety of real-world issues, such as classifying spam into separate inboxes. Methods used in this type of learning include neural networks, naive Bayes, linear regression, logistic regression, decision trees, and more.

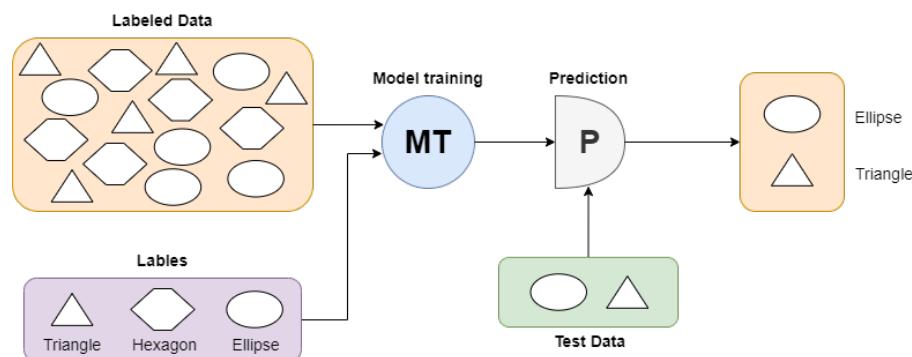


Figure 2. Supervised machine learning.

- Unsupervised learning—analyzes and aggregates untagged datasets using machine learning methods (see Figure 3). These algorithms discover hidden patterns or data clusters without requiring human participation. Unsupervised learning is suitable for exploratory data analysis, cross-selling techniques, consumer segmentation, and picture and pattern identification, as it can identify similarities and contrasts in information. In unsupervised learning, other algorithms include neural networks, k-means clustering, probabilistic clustering approaches, etc. For example, principal component analysis (PCA) and singular value decomposition (SVD) are two commonly used methods of unsupervised learning; they are used for dimensionality reduction and feature extraction.

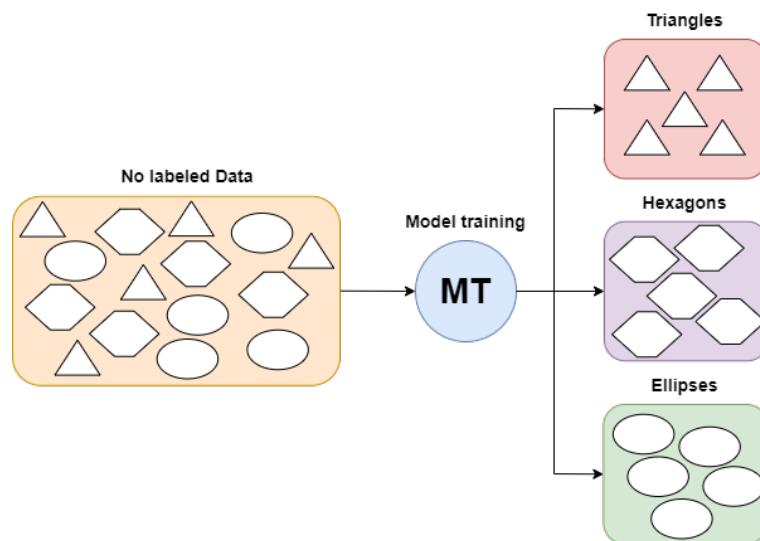


Figure 3. Unsupervised machine learning.

- Semi-supervised Learning—this represents the middle ground between learning with and without a teacher. During training, it uses a smaller set of labeled data to guide the classification and extraction of symptoms from a larger, unmarked dataset. Partially supervised learning (see Figure 4) can solve the problem of the lack of tagged data (or the inability to afford to tag enough data) to train a supervised learning algorithm [6].

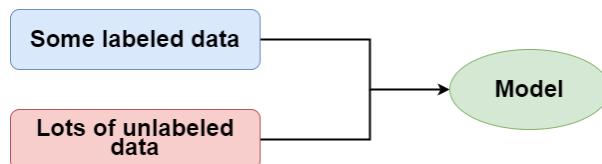


Figure 4. Semi-supervised machine learning.

- Reinforcement learning—this involves taking the proper steps to maximize compensation in a given circumstance. It is used by software and robots to determine the optimal behavior or course of action in a given circumstance. In supervised learning, the training data include the solution key, so the model is trained with the right answer alone, but in rewarded learning (see Figure 5), there is no response and no learning; the system determines how to complete the job. In the absence of a training set, the system is compelled to gain knowledge by experience [7].

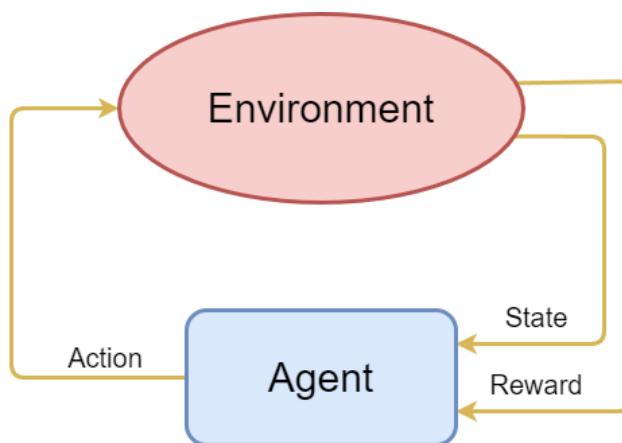


Figure 5. Reinforcement learning.

2.3. Deep Learning

Deep learning (DL) is a subset of machine learning (see Figure 1), which is a subset of artificial intelligence. Artificial intelligence is an umbrella term for approaches that enable computers to simulate human behavior. All of this is made possible by a collection of algorithms educated on data using machine learning [8].

Deep learning is a sort of machine learning influenced by the structure of the human brain. Deep learning algorithms attempt to reach comparable conclusions to those that would be reached by continuously analyzing data with a predetermined logical framework. To do this, deep learning employs multi-layered algorithmic structures known as neural networks (see Section 2.3.4).

2.3.1. Neural Networks

The design of the neural network (see Section 2.3.4) is based on the structure of the human brain. Neural networks may learn to recognize patterns and categorize data in the same way that the human brain does. Individual layers of neural networks may also be seen as filters that operate from coarse to fine, hence improving the likelihood of detecting and outputting the right results. The human mind functions similarly. When a person acquires new knowledge, the brain attempts to compare it to previously stored data. Deep neural networks use the same principle.

Neural networks provide a variety of operations, including grouping, classification, and regression. Using neural networks, it is possible to group or sort unlabeled data based on the similarities between the samples. In the case of classification, the network may be trained on a labeled dataset to categorize the samples within that dataset into several categories.

Neural networks can generally execute the same tasks as traditional machine learning techniques. However, they do not work in reverse. Deep learning models may conduct tasks that machine learning models are incapable of accomplishing due to the unique characteristics of artificial neural networks [8].

2.3.2. Biological Neural Networks

The neurons present in the human brain serve as the inspiration for artificial neural networks. In reality, artificial neural networks imitate some of the fundamental operations of neural networks in the human brain, although in a highly simplified manner.

A biological neural network is made up of many neurons. A typical neuron consists of a cell body, dendrites, and an axon (see Figure 6). Dendrites are thin, outgrowths of the cell body. Axons are cellular extensions that emanate from the cellular body. The majority of neurons receive signals through dendrites and transmit messages via axons.

Neuron

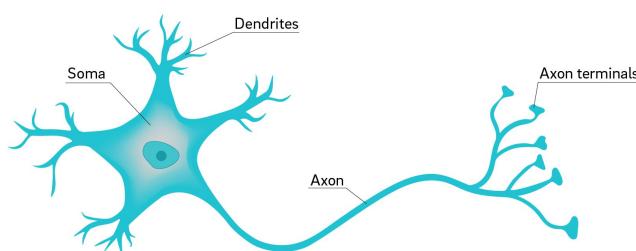


Figure 6. Biological neuron model [9].

In the majority of synapses, impulses go from the axon of one neuron to the dendrite. All neurons are electrically stimulated because their membranes maintain voltage gradients. If the voltage varies significantly and rapidly, the neuron creates an electrochemical pulse known as the action potential. This potential quickly propagates through the axon, activating synaptic connections as it reaches them [8].

2.3.3. Artificial Neural Networks

Typically, a neural network consists of a collection of linked units or nodes. We refer to these nodes as neurons. These synthetic neurons faithfully simulate the organic neurons of the human brain. A neuron is nothing more than a graphical representation of a number (e.g., 1.2, 5.0, 42.0, 0.25, etc.). Any link between two artificial neurons is equivalent to an axon in a biological brain. Scales, which are also nothing more than numerical values, are used to establish connections between neurons.

As an artificial neural network learns, the weights and the strengths of the connections between the neurons vary. The collection of weights for each job and dataset is unique. We cannot forecast the values of these weights; thus, they must be learned by the neural network. The process of learning is also known as training [8].

2.3.4. Architecture of Artificial Neural Networks

Neural networks are intricate structures consisting of artificial neurons that are capable of receiving numerous inputs and producing a single output. The primary function of a neural network is to convert input data into meaningful output data. Typically, a neural network consists of an input layer, an output layer, and one or more hidden layers in between.

All neurons in a neural network communicate with one another; therefore, they are all linked. The network is able to detect and observe every facet of a particular data collection, as well as how the various data elements may or may not be connected. In this manner, neural networks are able to identify exceedingly complicated patterns in massive datasets.

In a neural network, information flows in two directions:

- Forward neural network—in this paradigm, signals flow in just one way, from the input layer to the output layer. Power grids consist of an input layer, an output layer, and zero or more concealed layers. They are used extensively in pattern recognition.
- Recurrent neural network—in this paradigm, recurrent networks process a succession of inputs by using their internal states (memory). In these networks, signals may propagate in both ways via network loops (hidden layer/hidden layers). Typically, they are employed for time series and sequential activities [10].

2.3.5. Layer Interconnection in a Neural Network

As seen in (Figure 7), each connection between two neurons is represented by a unique weight, denoted by the symbol W . These weights W each have their own indices. The first value of the indices represents the number of neurons in the layer where the connection begins, while the second value represents the number of neurons in the layer where the connection terminates [8].

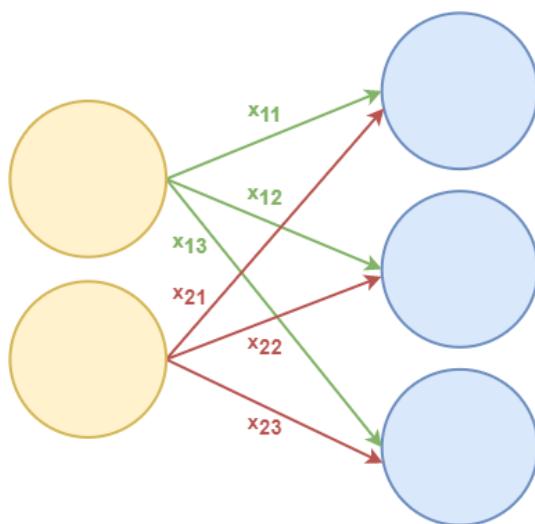


Figure 7. Connecting neuronal layers.

All weights between two layers of a neural network can be represented by a matrix called a weight matrix:

$$W = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{pmatrix} \quad (1)$$

There are as many entries in the weight matrix as there are neuronal connections. The size of the two layers that this weight matrix connects determines the dimensions of the weight matrix. The number of rows represents the number of neurons in the layer where the connections start, while the number of columns represents the number of neurons in the layer where the connections terminate.

In this instance, the number of rows of the weight matrix correlates to the size of the input layer, which is two, and the number of columns corresponds to the size of the output layer, which is three [8].

2.3.6. The Process of Learning the Deep Neural Network

In actuality, the learning rule (or learning process) involves mathematical logic. It applies this rule to the network and enhances the performance of an artificial neural network. Consequently, learning rules change a network's weights and distortion levels as they simulate in a certain data environment. The process of applying a learning rule is iterative. It improves the neural network's performance by allowing it to learn from existing situations.

The learning process can be divided into several types:

- Hebb's rule was the first rule of learning. In 1949, Donald Hebb developed it as a learning algorithm for an uncontrolled neural network. Hebb's rule of learning assumes that if two neighboring neurons are activated and deactivated at the same time, then the weight connecting those neurons should increase. In the case of neurons operating in the opposite phase, the weight between them should be reduced. If there is no signal correlation, the weight should not change. If the inputs of both nodes are either positive or negative, then there is a strong positive weight between the nodes. If the input of one node is positive and the other is negative, there is a strong negative weight between the nodes. Initially, the values of all weights are set to zero. This learning rule can be used for both soft and hard activation functions. Since the required neuronal responses are not used in the learning process, this is the rule of uncontrolled learning. Absolute weight values are usually proportional to the learning time, which is undesirable. The mathematical notation of Hebb's rule in neural network learning is as follows:

$$W_{ij} = x_i * x_j \quad (2)$$

- Perceptron learning rule—each connection in a neural network has an assigned weight that changes during learning. According to the example of supervised learning, the network begins its learning by assigning a random value to each weight. The output value is calculated on the basis of a set of records for which the expected output value can be known. This is a sample of learning that denotes the whole definition. As a result, it is called a learning sample. The network then compares the calculated output value with the expected value. It then calculates the error function, which can be the sum of the squared errors occurring for each individual in the learning sample [11]. The mathematical notation of perceptron learning rules in the neural network is as follows:

$$\sum_i \sum_j (E_{ij} - O_{ij})^2 \quad (3)$$

E_{ij} and O_{ij} represent the predicted and actual values of the j th unit for the i th person, respectively. The network then modifies the weights of the individual units and determines whether the error function has risen or reduced at each iteration. Similar to normal regression, this is a solution to the issue of least squares.

- Delta learning rule—this is one of the most prevalent learning principles. It requires guided instruction. This rule states that the change in the sympathetic weight of a node is proportional to a multiple of the mistake and input.

Mathematical notation of the Delta rule in a neural network [11]:

$$\Delta w = \eta(t - y)x \quad (4)$$

The output vector that corresponds to the right answer for a particular input vector is compared. If the difference is zero, there is no learning; otherwise, the algorithm changes its weights to lower it. The Delta learning rule may be used for both a single output unit and many output units. With this rule, one must presume that the mistake can be directly measured. The objective of using the delta rule is to minimize the error-causing disparity between the actual and predicted output [11].

- The correlation rule of learning is founded on a similar basis to Hebb's rule. Hebb hypothesizes that the weights between responding neurons should be more positive, whilst the weights between neurons with the opposite response should be more negative. The correlation rule, unlike Hebb's rule, involves supervised learning. In mathematical form, the correlation rule of learning is as follows:

$$\Delta w_{ij} = \eta x_i d_j \quad (5)$$

where d_j is the desired value of the output signal. This training algorithm usually starts with initializing the weights to zero [11].

- The Outstar learning rule is utilized when there is an assumption that nodes or neurons in the network are organized in layers. In this case, the scales linked to a certain node should correspond to the needed outputs for the neurons connected through these scales. Outstar provides the necessary response for the n -node layer. In mathematical form, Outstar learning is as follows [11]:

$$W_{jk} = \begin{cases} \eta(y_k - w_{jk}) & \\ 0 & \end{cases} \quad (6)$$

2.4. Machine Learning and Deep Learning

Deep learning (DL) and machine learning (ML) are currently the two most trendy technologies in the world. However, these technologies are often confused. Although deep learning is a subset of machine learning (see Figure 1), many people confuse these two terminologies. There are several distinctions between machine learning and deep learning, both in terms of resources and applications.

- Data consumption—DL requires a large number of labeled samples to be successful. However, the amount of data alone is not enough; they must be of the right quality, i.e., properly labeled. Not all data collected are flagged, labeled correctly, or in a manner appropriate for the DL. Such data are not always publicly available. In this case, data labeling needs to be done, which is time-consuming and costly and often requires a defined and rigorous set of procedures, quality control, and expertise. Unfortunately, this fact and its impact on the usefulness of DL for real problems are often downplayed in DL discussions.
- Specific hardware—the training phase of DL systems usually requires specialized hardware, such as graphics processing units (GPUs), in order to reduce the execution time to a manageable level, i.e., hours, days, or weeks, compared to years. These systems, although they are becoming cheaper, are still expensive compared to the needs of simpler ML kits.
- Specification extraction is the process of incorporating domain knowledge into the creation of extractors of individual specifications in order to reduce data complexity and make patterns for learning algorithms visible. This process is time-consuming and expensive. Figure 8 presents an example of a specification extraction difference between DL and ML.

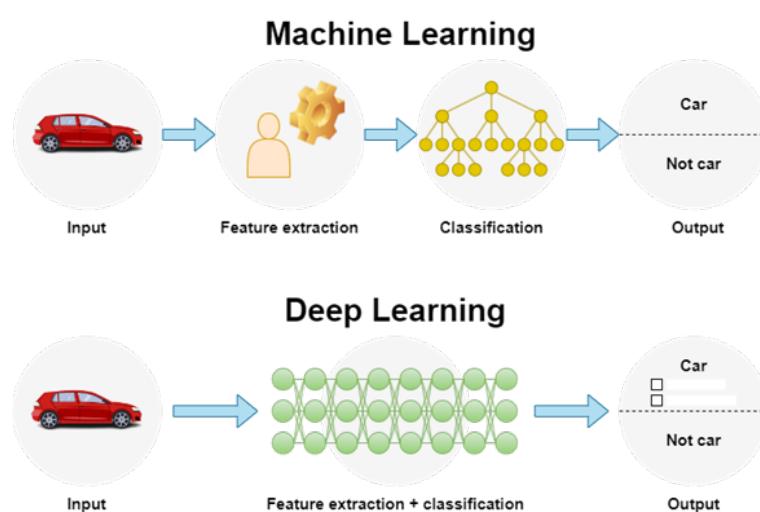


Figure 8. DL and ML operations process.

- Application—in terms of time, hardware, and data, DL is more costly. In conclusion, it can be claimed that deep learning and machine learning are most effective when employed in accordance with the conditions listed in Table 1 [12].

Table 1. Applications of ML and DL [13].

Criterion	Machine Learning	Deep Learning
Data volume	Fewer data	Large amounts of data
Computational costs	Shorter time and cheaper hardware.	Longer time and more expensive hardware.
Adaptability	To construct high-performance models, domain- and application-specific approaches, as well as specialized engineering, are required. Therefore, the resulting models are less flexible, even in comparable domains.	It is easier to adapt to different areas and applications.
Engineering tools	Complex specific engineering is often required, which is time-consuming as well as economical, due to the lack of experts in the field.	It may decrease or even remove the requirement for a detailed specification, considerably decreasing the time and expenses associated with this stage. However, it may incur greater hardware and time-related costs associated with deep learning procedures.
Interpretability	Due to specialized engineering and a simplified design, systems are often easy to comprehend. It is simpler to comprehend how and why the ML algorithm reaches its conclusion. This may be very helpful and necessary for updating and repairing a system that delivers inaccurate results under unforeseen conditions.	Less capable of interpretation. It is sometimes regarded as a “black box” system, where researchers attempt to explain how and why it produces a certain outcome. Nonetheless, major advancements continue to be made in this field, which exposes a black box, so that this difference may fade with time.

3. Libraries and Software for Machine Learning

Modern artificial intelligence systems and machine learning algorithms have revolutionized the approaches to scientific and technological challenges in various fields. Remarkable improvements in the quality of state-of-the-art computer vision, human language processing, speech recognition, and much more, can be observed. Although machine learning algorithms, deep learning algorithms, and individual architectural components can initially be expressed in mathematical notation, they must eventually be transcribed into a computer program for use in the real world. For this purpose, there are a number of software libraries (see Figure 9) and open-source machine learning programs, as well as commercial ones. MLC++ and OpenCV were the first libraries designed mainly for machine learning. In 2002, the Torch Library, which was intended for deep learning, appeared for the first time, i.e., in the field of deep learning [14].

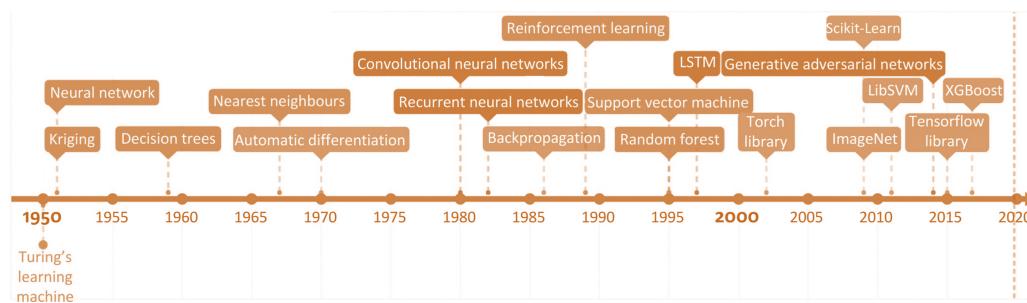


Figure 9. Publication of machine learning libraries on a timeline.

3.1. Programming Languages for Machine Learning

The most difficult aspect of mastering machine learning for beginners is deciding where to begin. Moreover, one of the most essential factors is selecting the proper programming language. Given that there are more than 700 commonly used programming languages, each with its own advantages and disadvantages, determining the best language for machine learning is a difficult endeavor. Table 2 displays the distribution of the most popular libraries (frameworks) and engineering software by the most popular programming languages that are suitable for machine learning and deep learning.

Table 2. Software libraries for AI sorted by programming language.

	Python	C++	Java	R	Lua	Scala	Own Language
Torch/PyTorch	• *				•		
TensorFlow	•	•					
Keras	•						
Caffe	•	•					
Deeplearning4j			•				
scikit-learn	•						
Apache Spark MLlib	•		•			•	
Apache Mahout			•			•	
Apache SINGA		•					
Shogun	•	•		•			
KNIME			•				
GNU Octave							•
MATLAB and Simulink							•
Maple							•
Wolfram Mathematica							•

* Label of programming languages for the given framework.

3.1.1. Python

Python is an object-oriented, high-level programming language with dynamic semantics; it is known for its interpretability. Its high-level built-in data structures, coupled with dynamic typing and dynamic binding, make it highly desirable for rapid application development, as well as for use as a scripting or linking language to connect existing components. Python's easy-to-learn syntax emphasizes intelligibility, thereby reducing the cost of program maintenance. Python supports numerous modules and packages, promoting the modularity and reusability of programs. The Python interpreter and comprehensive standard library are freely available in source or binary form for all main platforms and may be redistributed without restriction [15].

The Python code is known for its readability and simplicity as it is similar to the English language. This feature makes it easy to write and comprehend, reducing the time spent writing complex code for developers. Moreover, Python offers an extensive range of libraries and frameworks tailored to the needs of AI and machine learning (ML), such as computing capabilities, statistical computing, scientific computing, and more. This flexibility provides developers the opportunity to choose either OOPS or script-based programming. Python for machine learning provides a quick and easy display of results without the need for a complete recompilation of the code. Additionally, new developers who join the project midway can quickly understand the project's scope, minimizing the likelihood of errors and conflicting paradigms, leading to faster development of the machine learning program.

Data are crucial aspects of machine learning, artificial intelligence, and deep learning algorithms. They require intensive visualization to understand all of the variables and factors involved in the data analysis. Python offers developers with the opportunity to create histograms, charts, and graphs, providing a better understanding of how data interact and work together. Moreover, APIs facilitate the visualization process by enabling the generation of clear data reports.

There are four different styles of Python software that the user can choose from: imperative, object-oriented, functional, and procedural. These styles reduce the possibility of mistakes based on the artificial intelligence project [16].

3.1.2. C++

C++ is a widely recognized programming language extensively used by over four million programmers worldwide. It was first introduced in 1985 as an extension of the C programming language and has evolved into a distinct language with unique features and characteristics. It is an object-oriented and procedural language that provides discrete and direct memory management capabilities, making it ideal for large and scalable applications. C++ is considered a high-level programming language that is used in machine learning and deep learning applications.

As an object-oriented language, C++ enables developers to write powerful and efficient code that can run on a broad range of platforms, from small embedded systems to large supercomputers. At the machine level, C++ code is compiled into machine code that can be executed directly by the computer's processor, allowing for optimization of computing resources. In the context of machine learning, C++ is commonly used to implement low-level algorithms, such as matrix operations and optimization algorithms, which require high-performance computation. C++ is also utilized to interface with specialized hardware, such as GPUs, to accelerate computations.

In summary, C++ is a versatile and powerful programming language that can be applied to various applications, including machine and deep learning. Its low-level access to hardware and optimization for specific architectures makes it an appealing option for computationally intensive tasks [17].

3.1.3. R

R is a programming language that has been specifically designed for statistical computing and data analysis. It offers an array of libraries and tools for data manipulation, modeling, and visualization, making it a top choice for tasks related to machine learning. The language is widely known for its comprehensive collection of packages that cater to machine learning applications. For instance, the "caret" package provides an all-inclusive set of tools for building and assessing machine learning models, while the "randomForest" package offers implementation of the random forest algorithm. R is also celebrated for its interactive data visualization capabilities, which simplify the exploration and comprehension of intricate datasets. Developers commonly use the "ggplot2" package to create high-quality, personalized visualizations. The language boasts of an active community of users and developers who regularly contribute to the development of packages and share their knowledge through various online forums and resources, making it easy for beginners to get started with machine learning in R. R supports popular machine learning algorithms, such as linear regression, logistic regression, decision trees, random forests, support vector machines, and neural networks, which can be used for a variety of applications, including classification, regression, clustering, and dimensionality reduction. In summary, R is a robust and adaptable programming language that is well-suited for machine learning tasks, thanks to its broad range of packages, interactive visualization capabilities, and active community [18].

3.1.4. Java

Java is a commonly used programming language that is widely used in both industry and academia to create a variety of applications, including those related to machine learning. Its popularity stems from its simplicity, portability, and security, making it a sought-after option for developing sturdy and scalable applications.

In the field of machine learning, Java is frequently employed to create and implement large-scale applications, such as fraud detection systems, recommender systems, and natural language processing tools. It is equipped with numerous libraries and frameworks for machine learning, including Weka, Deeplearning4j, and Apache Mahout. Among these, Weka is a popular open-source machine learning library in Java that includes an extensive set of algorithms for data mining, classification, and feature selection. Additionally, it offers tools for preprocessing, visualizing, and evaluating data, making it straightforward to create and assess machine learning models. Deeplearning4j, another well-known deep learning framework written in Java, can run on distributed systems, such as Hadoop and Spark, providing a variety of neural network architectures and optimization algorithms, including convolutional neural networks, recurrent neural networks, and gradient descent. This makes it an excellent option for a wide range of deep learning tasks. Lastly, Apache Mahout, a popular machine learning library in Java, contains various algorithms for clustering, classification, and collaborative filtering; it is designed for scalability and is capable of operating on distributed systems such as Hadoop and Spark, making it a good choice for processing and analyzing vast amounts of data.

Overall, Java is a versatile and potent programming language that can be utilized for a broad range of machine learning applications. Its simplicity, portability, and security make it a popular option for developing and deploying robust and scalable applications in both industry and academia. [19]

3.2. Libraries (Frameworks) for Machine Learning

A machine learning framework is an interface, library, or tool that enables developers to generate machine learning models with less effort and in less time by abstracting away the intricacies of fundamental methods. The library offers a straightforward and quick method for defining machine learning models by utilizing a collection of pre-built, optimized components. Some of the key features of a good machine learning framework are:

- It is optimized for high performance;
- It is developer-friendly, i.e., the library uses traditional modeling methods;
- It is easy to understand and code;
- It is not completely a black box;
- It provides parallelization to the computational process distribution.

An effective machine learning library simplifies the process, making it accessible to a greater number of developers [20]. Table 2 presents the distribution of software libraries across different programming languages. Some of these programs have multiple language APIs, so users can choose the one that best fits their needs.

3.2.1. Torch/PyTorch

Torch is a platform for scientific computing that provides comprehensive support for machine learning methods. It is built for deep Lua-based learning and is extensively used by Facebook, Twitter, and Google. It employs CUDA code for processing together with C/C++ libraries and was designed to expand the production and general flexibility of constructing models. PyTorch, unlike Torch, runs on Python, allowing anybody with a basic understanding of Python to begin building their own deep learning models. PyTorch is essentially a port of the Torch deep learning library, which is used to build deep neural networks and perform sophisticated tensor computations [21].

3.2.2. TensorFlow

Unquestionably, TensorFlow is one of the most popular deep learning frameworks. It was developed by the Google Brain team and supports languages such as Python, C++, and R, in addition to packaging libraries for the creation of deep learning models. It is accessible on both desktop and mobile platforms. Google Translate, in combination with features such as natural language processing, text categorization, summarization, audio/image/font recognition, prediction, and labeling, is the most well-known use of TensorFlow. TensorBoard consists of a set of visualization tools for TensorFlow, which allow for efficient viewing of network modeling and performance data. Another TensorFlow utility, TensorFlow Serving, is used to rapidly deliver new algorithms/experiments while preserving the same server architecture and API. It also offers integration with other TensorFlow models, extending beyond standard methods, and can be modified to accommodate additional models and data formats. TensorFlow is one of the most popular deep-learning frameworks due to the fact that it is based on the Python programming language, is maintained by Google, and offers user-friendly documentation and instructions [21].

3.2.3. Keras

The Keras Library was designed with quick experimentation in mind. It was developed in Python and enables running convolutional and recurrent networks on TensorFlow or Theano. The Keras deep learning framework was created to provide a simpler interface for quick prototyping by generating active neural networks that are compatible with TensorFlow since the TensorFlow interface is sophisticated and may be difficult for novice users. In summary, Keras is lightweight, user-friendly, and minimalist. These are the reasons why Keras is included in the TensorFlow basic API. The primary applications of Keras include text categorization, generation, summarization, labeling, translation, and voice recognition, among others [21].

3.2.4. Caffe

Caffe is renowned for its laser-like velocity. It is a deep learning framework that supports interfaces, such as C, C++, Python, MATLAB, and the command line. In recent years, it has gained popularity due to its usefulness in convolutional neural network (CNN) modeling and its speed. Having access to the “Caffe Model Zoo” deep network repository is a major benefit of utilizing the Caffe library in C++. Caffe Model Zoo provides nets that are already trained and ready for use. This library must be designed for CNN modeling and image processing problem-solving. The greatest benefit of Caffe is its quickness. Using a single Nvidia K40 graphics processor, it can process over sixty million photos each day. Caffe is also a popular network for visual recognition. However, it does not support network layers with precise granularity, such as TensorFlow or CNTK. Due to the design, support for recurrent networks and language modeling is rather minimal, and complicated layer types must be created using a low-level language [21].

3.2.5. Deeplearning4j (DL4J)

The letter j in Deeplearning4j stands for Java. It is a deep learning library for the Java Virtual Machine (JVM). It is written in Java and supports Scala, Clojure, and Kotlin, among other JVM languages. Important aspects of the Eclipse Deeplearning4j deep learning framework include parallel training through iterative reductions, adaption of the microservice architecture in combination with distributed CPUs and GPUs, and microservice architecture adaptability. Widely recognized as a commercial, industry-oriented, and distributed deep learning platform, DL4J provides extensive support for deep networks including RBM, DBN, CNN, RNN, RNTN, and long-term short-term memory (LSTM). Because this framework for deep learning is developed in Java, it is far more effective than Python. DL4J is just as fast as the Caffe framework for multi-GPU image recognition applications. The picture identification, fraud detection, text mining, audio tagging, and natural language processing capabilities of this library are unrivaled [21].

3.2.6. scikit-learn

scikit-learn is a machine learning package for Python. Specifically, it consists of a collection of simple and useful tools for data mining and analysis, as stated by the creators. The framework is composed of numerous well-known Python programs, including NumPy, SciPy, and matplotlib. This library's primary benefit is the BSD license under which it is provided. This license enables the user to select whether or not to stream their modifications without limiting commercial usage. The primary benefits of this solution are its accessibility and simplicity; moreover, it is simple to use, even for novices, and is an excellent option for straightforward data analysis jobs. On the other hand, scikit-learn is not the greatest option for deep learning [22].

3.2.7. Apache Spark MLlib

MLlib is an Apache Spark-integrated machine learning library. It is a tool used for processing massive volumes of data and is sometimes referred to as an open-source cluster computing platform. It operates on numerous platforms, including EC2, Hadoop YARN, Mesos, and Kubernetes, and is completely compatible with the NumPy library and the R programming language. MLlib is capable of accessing data from HDFS, Apache Cassandra, Apache HBase, Apache Hive, and many more data sources. In terms of its benefits and drawbacks, its advantages include highly rapid processing, dynamism, reusability, and fault tolerance. However, weaknesses, such as resource needs, excessive latency, the necessity for manual optimization, and inadequate file management might degrade the user experience of MLlib. It is most useful for fraud detection and e-commerce data management applications [22].

3.2.8. Apache Mahout

Apache Mahout is a scalable machine learning (ML) toolkit in distributed data flow systems that implements classification methods, clustering, dimension reduction, and recommendations in a variety of ways. In 2008, when it began to concentrate on MapReduce, which was the leading abstraction for scalable computing at the time, Mahout was a pioneer in large-scale machine learning. Mahout is extensively used by industry-leading online organizations and is included in a number of commercial cloud services. In recent years, Mahout has transitioned to a framework that permits a mix of data flow programming and linear algebraic computations using backends such as Apache Spark and Apache Flink. This approach enables users to conduct model preprocessing and training in a single unified dataflow system, as opposed to requiring the integration of many specialized systems. Mahout is an open-source project maintained by the Apache Software Foundation [23].

3.2.9. Apache SINGA

By partitioning the model and parallelizing the training process, Apache SINGA focuses largely on distributed deep learning. It offers a simple and dependable programming approach that is capable of operating on a cluster of nodes. Principal applications are in image recognition and natural language processing (NLP). SINGA was designed with an easy programming approach based on layer abstraction and supports several deep learning models. Based on a highly adaptable architecture, it is capable of running synchronous, asynchronous, and hybrid training techniques. Three essential components comprise the SINGA technological stack: IC, model, and core. The IO component comprises classes used for network and disc read/write operations. The kernel component is responsible for tensor operations and memory management. The model component comprises machine learning model-specific algorithms and data structures [24].

3.2.10. Shogun

Shogun is one of the oldest and most esteemed machine learning libraries; it was founded in 1999 and built in C++, although it is not confined to that language. Shogun may be used transparently in languages and settings, such as Java, Python, C#, Ruby, R,

Lua, Octave, and MATLAB thanks to the SWIG library. Shogun is intended for unified, complete learning across a broad variety of learning criteria and environments, including classification, regression, and exploratory data analysis [25].

3.3. Commercial Software for Machine and Deep Learning in Mechanical Engineering

Currently, computers learn or discover without explicit programming how to accomplish things. However, all algorithms need the requisite software to function effectively. This enables individuals to teach computers to execute tasks using huge volumes of data, where the computers learn to identify patterns, characteristics, and trends that enable them to make judgments and predictions. Today, machine learning software learns from the large quantity of accessible web data. In addition to collecting data, the finest machine learning technologies can contextually analyze and categorize the content, sentiments, and other subtleties of datasets [26].

3.3.1. GNU Octave

Octave is a high-level programming language designed for numerical calculations. It can quickly solve linear and nonlinear numerical problems and perform other experiments numerically. The Octave language is quite similar and mostly compatible with MATLAB. If the code runs in MATLAB without using functions that Octave does not have, it will also work in Octave. It even has several language functions and syntax diversity that MATLAB lacks. Unlike MATLAB, it is public and completely free. Although Octave was originally designed for scientific computing, many organizations use it for basic data processing and graphing. It can also be used for machine learning, data analysis, and the creation of ML algorithms [27].

3.3.2. MATLAB and Simulink

MathWorks' MATLAB is a high-performance programming language used for technical computations. It combines computations, visualizations, and programming in a user-friendly environment, where problems and answers are stated in a familiar mathematical language.

MATLAB is an interactive system; its fundamental data element is a dimensionless array. This enables one to solve many complex computing issues in a fraction of the time it would take to build a program in a scalar non-interactive language, such as C or Fortran.

The application-specific solutions in MATLAB are known as toolboxes. For the majority of MATLAB users, toolboxes that enable the study and application of specialist technologies are crucial. Toolboxes are exhaustive sets of MATLAB functions (M-files) that enhance the capabilities of the MATLAB environment to address certain types of problems. Toolboxes are provided for a variety of disciplines, including signal processing, control systems, neural networks, fuzzy logic, wavelets, and simulation. Moreover, one of the greatest benefits of this application is that it has an online version, which means it can operate in any web browser [28].

3.3.3. Maple

Maple is a symbolic and numeric computing environment as well as a multi-paradigm programming language. It includes symbolic mathematics, numerical analysis, data processing, and visualization, among other fields. The MapleSim toolkit includes multi-domain physical modeling and code-generating capabilities. Maple's symbolic computation capabilities are comparable to those of a universal computer algebra system. For instance, it is able to manipulate mathematical expressions and develop symbolic solutions to specific problems, including those derived from ordinary and partial differential equations. It can also use the toolkit for deep learning. Maple is developed commercially by Maplesoft, a Canadian software firm [29].

3.3.4. Wolfram Mathematica

Wolfram Mathematica is a software system with built-in libraries for several areas of technical calculations, allowing users to perform symbolic calculations, manipulate matrices, plot functions and different data types, implement algorithms, create user interfaces, and establish connections with programs written in other programming languages.

Wolfram Language provides cutting-edge designs, training, and deployment capabilities for neural network machine learning systems. There are several common kinds of layers that are combined symbolically into a network, which can then be trained and deployed instantly on available CPUs and GPUs. The Wolfram Mathematica online version, which can operate in any web browser, is one of the program's greatest features [30].

3.3.5. Other Commercial Software

In addition to the commercial software mentioned in Section 3.3, there are many others on the market. Table 3 lists the most well-known companies that have products in their portfolios that can be commonly used in mechanical engineering for mechanical design and optimization. However, in order to be competitive, all brands are attempting to implement tools to work with machine learning and artificial intelligence in their software.

Table 3. Software companies that offer products that use artificial intelligence.

Software Company	Software Company
ANSYS [31]	ESI [32]
Altair [33]	Materialise [34]
HEXAGON [35]	NUMECA [36]
Dassault Systèmes [37]	Rhino [38]
PTC [39]	COMSOL [40]
Autodesk [41]	IDEA StatiCa [42]
Siemens [43]	Noesis [44]
AVEVA [45]	BETA-CAE Systems [46]

4. Method of Representing Input Data for AI

This section covers the prevalent techniques used to represent designs in design graphical models for engineering. It contains an explanation of each approach, a broad overview, and the advantages and disadvantages associated with each method.

Input data are important components in machine learning. They denote the observations or measurements that can be used to train a machine learning model. The quality and quantity of data available for training and testing play important roles in determining the performance of a machine learning model. Data can take different forms, such as numerical, categorical, or time series, and can come from a variety of sources, such as databases, spreadsheets, or APIs. Machine learning algorithms use data to learn patterns and relationships between input variables and target outputs, which can then be used for prediction or classification tasks.

4.1. Images

Design information can be in the form of images, such as microstructure scans or topology optimization. An image is a rectangular grid of pixels, with each pixel having a color parameter represented by a third-order tensor with the height, width, and channels. Common color schemes include black-and-white, grayscale, and color. The advantages of using images are that they are information-rich and can capture many design details, and deep learning filters can generate both high- and low-level features, as well as up-sampling and down-sampling. However, representing designs using pixels can result in generating images that are impractical for downstream tasks, and fabricating designs accurately from images can be difficult. Even performance evaluations using simulation tools may require an intermediate conversion from the image to the 3D model. Artifacts are also common in image-based designs, especially when training on small datasets. Images may

lack domain knowledge and physical realization information, leading to a gap between the generated images and the actual designs they represent in design graphical models.

4.2. Voxelization

To represent 3D designs, voxels are often used, which are the 3D counterparts of pixels. Voxelizations have similar features to images but are typically represented as Boolean values (space vs. object) instead of color parameters. This makes them third-order tensors with dimensions consisting of height, width, and depth. The advantages of voxelizations are that they support 3D convolution and can learn both high-level and low-level features in 3D. However, the curse of dimensionality is more pronounced with voxelizations than with images, with the number of parameters increasing with the cube of spatial resolution. Similar to images, voxelizations are limited in their usability in downstream tasks and suffer from prevalent artifacts. They are often used as surrogate representations of CAD models or 3D shapes and require conversion before they can be used in downstream tasks. For example, they are typically converted to boundary representations or polygonal representations, which serve as the native parameterizations used in rendering, graphics software, finite element analysis, and computational fluid dynamics simulation.

4.3. Point Clouds

Point clouds are sets of points in 3D space that define the shape of an object. They have several advantages, including the ability to represent complex geometry with a finite number of points, and they are easy to create using 3D scanning software. However, point clouds also have some drawbacks, including the need to convert them to other representations, such as meshes for downstream tasks, which can be challenging [47,48].

4.4. Meshes

Meshes are frequently used to represent objects in 3D spaces; triangular meshes are the most commonly used forms. They are the native representations used in many computer graphics algorithms and software, as well as finite element analysis (FEA) and computational fluid dynamics (CFD) tools. Meshes can be easily visualized and simulated in many FEA or CFD tools, which makes it easy to evaluate their performance using numerical methods. Meshes can be considered specialized types of graphs and can use graph convolutional operators. However, generating meshes directly using machine learning methods is more challenging compared to other representations, such as voxelizations and point clouds, despite recent advances in algorithms that can directly generate meshes [49–51].

4.5. Signed Distance Function

The signed distance function/field (SDF) is a technique used for parameterization, which involves a map from a coordinate point to an SDF value, usually in 3D. This value represents the distance from the point to the closest surface of the object and indicates whether the point is inside or outside the object. SDFs can be represented in different ways, such as rasterized representations, where each voxel contains a continuous numerical value, indicating the SDF value at that point. SDFs are useful as intermediate parameterizations for many learning tasks. However, similar to point clouds and voxels, SDFs require conversion to BRep or polygonal representations before they can be used in downstream tasks.

4.6. Parametric

In simple terms, “parametric” data refer to design representations that consist of design parameters without any known spatial or temporal significance. These parameters can be organized in a tabular format where each row represents a design and each column describes a design parameter. The advantages and disadvantages of using parametric data may vary depending on the specific case.

Pros: Parametric data are usually very information-dense, which means that they can encode more geometric details using fewer parameters. This can make the optimization

of parametrically represented designs much easier. Parametric data can also support downstream tasks, especially if the design parameters are human-interpretable. This can allow generated designs to be directly fabricated using conventional manufacturing techniques. Additionally, linking design parameters with latent variables can allow for more effective optimization or inverse designs using generative methods.

Cons: Learning parametric data can be difficult since it commonly involves mixed data types and inherits the training challenges of its components. Multimodal distributions, skewed categories, non-Gaussian distributions, data sparsity, and poor data scaling can make the application of DGMs and training very challenging. Since methods that are robust to all of these challenges are hard to come by, successfully applying existing methods to the parametric data domain can be challenging. Finally, since parametric data may be nontrivial to convert to 3D models, generated parametric designs may be difficult to evaluate using numerical simulations or through qualitative visualization [52,53].

4.7. Grammar

Grammar is a type of design representation that includes variables, terminal symbols, non-terminal symbols, and a set of rules that define how non-terminal symbols can be expanded into other symbols. Graph and spatial grammar are the most commonly used types of grammar in engineering design and have been applied in a wide range of applications, such as satellite and electromechanical system design. Grammar can be useful for controlling the assembly hierarchy of design components to ensure feasibility. The main advantage of using grammar is that it can explicitly constrain design spaces based on domain knowledge. However, a major disadvantage is that grammar can be difficult to learn implicitly and may need to be manually defined. Additionally, in some cases, grammar may restrict the exploration of the entire design space [54,55].

4.8. Graphs

In design, graphs are flexible representation methods composed of nodes and edges that can be directed or undirected. They have been successful in various design aspects and offer efficient ways to describe complex systems. The benefits of using graphs include their adaptability and capability in representing different kinds of designs and processes; they model complex interactions in systems and allow for the automation of system designs or inter-part dependency modeling. Graph neural networks (GNNs) are excellent tools for machine learning on graphs. On the other hand, despite the success of GNNs in molecular graph generation, there is less usage of graph-based design graphical models in designs, possibly because of the limited availability of graph-based design datasets [56,57].

5. Datasets

Datasets are some of the key components of machine learning, as they provide the raw data that machine learning algorithms use to learn patterns and make predictions. A dataset consists of a collection of data points that share some common characteristics, and machine learning algorithms use these datasets to learn patterns and make predictions. Datasets can come in many different forms and can be used for a wide variety of applications. Some datasets are generated from sensors or other data collection devices, while others may be scraped from websites or compiled from other sources. The quality of a dataset is important to the accuracy of the predictions made by machine learning algorithms; as such, data cleaning and pre-processing are essential steps in the data analysis process. Machine learning algorithms can be supervised or unsupervised, depending on whether the dataset includes labeled or unlabeled data. In supervised learning, the dataset includes labeled data, where each data point has a known outcome or label, and the algorithm learns to predict the outcome based on the input data. In unsupervised learning, the dataset includes unlabeled data, and the algorithm learns to identify patterns and relationships in the data without any known outcomes or labels. It is important to choose the right dataset for the specific machine learning task at hand. The dataset should be representative of the problem

being solved and should be large enough to provide enough data for the algorithm to learn from. In addition, the dataset should be diverse enough to cover a wide range of scenarios and situations, to ensure that the algorithm can generalize to new situations. Finally, it is important to properly evaluate the performance of a machine learning algorithm when using the chosen dataset. This involves splitting the dataset into training and testing sets, and using the training set to train the algorithm and the testing set to evaluate its performance. The evaluation metrics used will depend on the specific problem being solved, but commonly used metrics include accuracy, precision, recall, and F1 score [58–60].

5.1. Images and Videos Datasets

Video and image datasets are collections of visual data that are used for training machine learning models. These datasets can come in various sizes and formats, and they can be used for a wide range of applications, from object recognition and tracking to facial recognition and emotion detection.

One of the most well-known video and image datasets is ImageNet [61], which contains millions of images labeled with over 20,000 categories. ImageNet has been used to train a variety of deep learning models, including convolutional neural networks (CNNs) for image recognition tasks.

Other popular video and image datasets include COCO (Common Objects in Context) [62], which contains images labeled with object categories and segmentation masks, and Open Images, which contains millions of images with annotations for object detection and segmentation.

In addition to these general-purpose datasets, there are also many specialized datasets for specific applications. For example, the Labeled Faces in the Wild (LFW) dataset consists of face images that have been widely used for facial recognition research, and the Kinetics dataset is a large-scale video dataset designed for action recognition.

Another image and video datasets:

- CIFAR-10 and CIFAR-100 [63]: These are datasets consisting of small images with 10 and 100 classes, respectively, and they are often used as benchmarks for image classification models.
- MNIST [64]: A dataset of hand-written digits used to benchmark image classification models.
- Pascal VOC [65]: A dataset of images with 20 object classes and segmentation masks; it is used for object detection and segmentation.
- Kinetics [66]: A large-scale dataset of human action videos with labeled action classes. It is used to train models for action recognition and detection.
- UCF101 [67]: A dataset of human action videos with 101 action classes; it is used for action recognition and detection.

5.2. CAD and CAD-Based Datasets

Computer-aided design-based datasets are collections of data that include CAD models, CAD files, and other design-related data used in computer-aided design applications. These datasets are used for a wide range of applications, including 3D modeling, product design, architecture, and engineering.

CAD datasets can come in different formats, such as AutoCAD, SolidWorks, or CATIA, and can be used to create models for different purposes. For instance, CAD data can be used for creating digital prototypes, virtual testing, and simulations. CAD-based datasets are particularly useful for machine learning applications, such as generative design, shape optimization, and design automation.

Some of the popular CAD-based datasets include:

- ShapeNet [68]: ShapeNet is a large-scale dataset of 3D models that covers a wide range of object categories, including furniture, vehicles, and animals. The dataset contains over 55,000 3D models, which can be downloaded in various formats, including OBJ, PLY, and OFF.

- ModelNet [69]: It contains over 127,000 3D CAD models from 662 object categories, which are uniformly aligned and normalized to a unit sphere. The dataset is split into training and testing sets, and the models are annotated with class labels for supervised learning.
- CAD-60 and CAD-120 datasets [70]: The CAD-60 and CAD-120 datasets contain 60 and 120 CAD models, respectively, which cover a wide range of object categories, including furniture, vehicles, and household items. The models can be downloaded in various formats, including SolidWorks and AutoCAD.
- McMaster-Carr [71]: McMaster-Carr is a platform that provides free 3D CAD models of mechanical components, such as bearings, screws, and gears. The platform has a large collection of models that can be downloaded in various formats, including SolidWorks, AutoCAD, and Inventor.
- TraceParts [72]: TraceParts is a platform that provides free 3D CAD models of industrial components, such as pumps, motors, and valves. The platform has a vast collection of models that can be downloaded in various formats, including SolidWorks, AutoCAD, and CATIA.
- GrabCAD [73]: GrabCAD is a community-driven platform that provides free 3D CAD models and designs that one can download and use for projects. The platform has a large collection of CAD files in various formats, including SolidWorks, AutoCAD, and CATIA.
- 3D ContentCentral [74]: 3D ContentCentral is a platform that provides free 3D CAD models of industrial components, such as gears, bearings, and motors. The platform has a large collection of models that can be downloaded in various formats, including SolidWorks, AutoCAD, and Inventor.
- PARTcommunity [75]: PARTcommunity is a platform that provides free 3D CAD models of industrial components, such as pumps, valves, and cylinders. The platform has a vast collection of models that can be downloaded in various formats, including SolidWorks, AutoCAD, and CATIA.
- Thingi10K [76]: This is a dataset comprising 10,000 3D-printing models. These models are sourced from the featured “things” on thingiverse.com, and suitable for testing 3D-printing techniques, such as structural analysis, shape optimization, and solid geometry operations.
- ABC [77]: This is a big CAD model dataset for geometric deep learning. It is a dataset used for geometric deep learning, consisting of over 1 million individual (and high quality) geometric models, each associated with accurate ground truth information, including the decomposition into patches, explicit sharp feature annotations, and analytic differential properties.
- Fusion 360 Gallery Dataset [78]: The Fusion 360 Gallery Dataset contains rich 2D and 3D geometry data derived from parametric CAD models. The reconstruction dataset provides sequential construction sequence information from a subset of simple ‘sketch and extrude’ designs. The segmentation dataset provides a segmentation of 3D models based on the CAD modeling operation, including B-Rep format, mesh, and point clouds.

5.3. Tables and Text Datasets

- Gas turbine CO and NO_x emission dataset [79]: This dataset contains 36,733 instances of 11 sensor measures aggregated from over one hour (by means of the average or sum) from a gas turbine located in Turkey’s northwestern region, for the purpose of studying flue gas emissions, namely CO and NO_x (NO + NO₂).
- Turbofan jet engine dataset [80]: This dataset is the Kaggle version of the very well-known public dataset for asset degradation modeling from NASA. It includes run-to-failure simulated data from turbofan jet engines.
- Bearing dataset [81]: This dataset consists of data from an experiment that involved a shaft with four bearings. Vibration data were collected. The dataset contains three

test-to-failure experiments, and each dataset consists of multiple files containing one-second vibration signal snapshots taken at specific intervals.

6. Machine Learning in Mechanical Design and Optimization

Machine learning, by definition, refers to the development of knowledge by computers using input data. Mechanical engineering is a strong area for this technology. Older machine builders have gained significant knowledge through several case studies and efficient workflows. This creates the opportunity to approach new projects, to be able to either use their technical knowledge with familiar elements that have already been discovered or use their experience to learn and adapt to new challenges. Newcomers to the field learn in this way and gather their effectiveness and knowledge; when it comes to learning, engineering is one of the areas where much of the learning takes place at work. How well it pays depends on the ability to collect and process information.

6.1. Product Design

Machine learning can improve the way machines and equipment are designed in many ways, e.g., by combining different parallel analyses of physics, solid mechanics, fluid mechanics, and so on. Using neural networks, it is possible to force a computer to learn to distinguish between parts and different assemblies and tools, which can be used in the future for designing a wide range of components and for data entry purposes in the mass production of various products. The following subchapters provide more detailed research projects from around the world, which deal with integration, respectively, using artificial intelligence and machine learning in the process of mechanical construction and design.

6.1.1. Platform for Integrated Aircraft Design

Efforts to shorten aircraft design times have stimulated the development of collaborative frameworks that facilitate the design of aircraft and their systems in a more integrated manner. The objective of the project was to create an integrated modeling and simulation framework that combines a filtering process to reduce the number of feasible architectures, a modeling platform that simulates an aircraft's power system, and a machine learning-based clustering and optimization module. This framework enables the designer to prioritize several designs and provides traceability for the best choices. In addition, it allows the integration of models at several degrees of realism, depending on the size of the design area and the desired precision. It exhibits the use of different electrical control technologies to electrify the main flight control system (PFCS) and the landing gear braking system. Key performance indicators are used to assess the performance of various architectures (fuel consumption, weight, performance). The optimization procedure employs a data-driven localization stage to recognize files with comparable structures. The framework displays the capacity to optimize across numerous system architectures in a manner that is efficient and scalable for bigger design spaces and problems of greater sizes [82].

6.1.2. Investigate Models for Efficient Decoding of 3D Point Clouds

This study focuses on surrogate modeling strategies to learn the approximation of computationally demanding 3D model function assessments. In the past, 3D point clouds were considered too large as data formats for surrogate modeling. However, employing neural network advancements to automatically encode 3D objects, these point clouds may now be translated to one-dimensional latent spaces [83]. This gives rise to a fundamental research question: what alternative modeling approach is best for discovering the links between the 3D geometric attributes of objects represented in a coded latent vector and the physical events collected in the assessment software? Unintuitively, it has been shown that decoding latent representations of 3D objects into performance predictions is far more successful than using a neural network decoder. Surrogate models without neural networks obtain equivalent accuracy to neural network models in test instances, including 3D model airplane datasets and watercraft datasets (see Figure 10). A test case confirmed

the equivalent accuracy of the modeling methodologies, but it was also discovered that the distribution of data performance values, particularly the existence of numerous outliers, had a considerable negative influence on accuracy. These findings challenge the commonly held belief that neural networks provide an effective “universal” solution for learning black functions and imply that even in systems that use several neural networks, possibly more efficient solutions for each network should be examined. Depending on the needed application accuracy, this surrogate modeling technique might be used to simulate costly simulation software, or if the error tolerance is low, it can be used as a first step to restrict the number of conceptual designs for further examination [84,85].

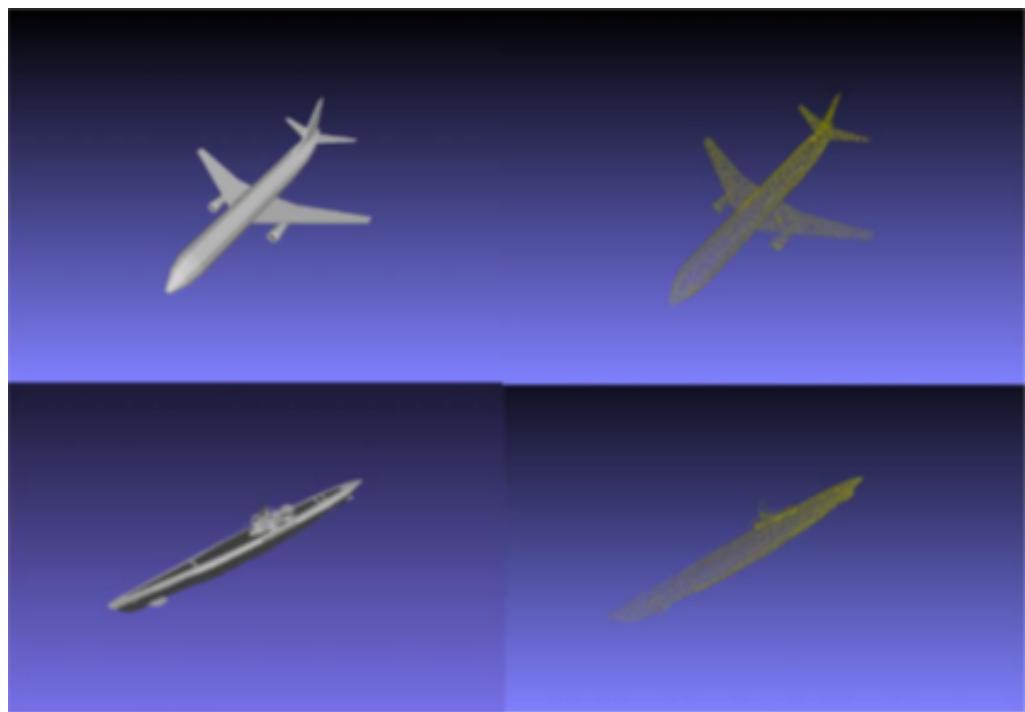


Figure 10. Demonstration of networks and corresponding point cloud models from aircraft (**top**) and vessel (**bottom**) datasets [85].

6.1.3. Machine Learning of Object Shapes through 3D Generative-Adversarial Modeling

In this study, a novel framework, the 3D generative inverse network (3D-GAN), which creates 3D objects (see Figure 11) from probability space utilizing recent breakthroughs in volume convolutional networks and generative reverse networks, was suggested. The proposed model has three advantages: first, the use of an inverse criterion instead of traditional heuristic criteria enables the generator to implicitly capture the structure of the object and enables the synthesis of high-quality 3D objects; second, the generator creates mappings from low-dimensional probability space to 3D object space, making it possible to sample objects without a reference image or CAD model and examine the diversity of 3D objects; and third, the proposed model is applicable to a wide range of applications. Experiments demonstrate that the suggested technique creates high-quality 3D models, and that the learned attributes (unsupervised machine learning) achieve excellent performance in identifying 3D objects, equivalent to the performance of supervised learning methods [86].



Figure 11. The 3D reconstruction of images on a dataset from IKEA [86].

6.1.4. Design of a Spatial Recurrent Neural Network for Design Form Optimization

In this study, a novel strategy for optimizing the structure and behavior of complex systems was devised. The method employs spatial grammar contained in character-recurrent neural networks (RNNs) to define the system, including the number of actuators and degrees of freedom, reinforcement learning to optimize actuator behavior, and physical simulation systems to assess performance and provide data for RNN (re)training. Grammar and RNN allow for a more complicated, combinatorically limitless design space as compared to the parametric optimization of a design with a fixed number of inputs. In the suggested technique, the RNN is first taught the spatial language defining the assembly arrangement, component geometry, material attributes, arbitrary numbers, and actuator degrees of freedom. The produced designs are further assessed in a physics-based environment, with the internal optimization loop using improved learning to identify the optimal actuator control strategy [87]. An RNN equipped with high-performance grammar can generate a design that is optimized in terms of both shape and behavior. Two assessment case studies based on the design of a modular sailing vessel (see Figure 12) are provided. The first case study optimizes the design without control surfaces, enabling RNN to comprehend the rationale behind high-performance solutions. The second case study expands the first to include controllable actuators, necessitating improvement of the inner loop's behavior [55].

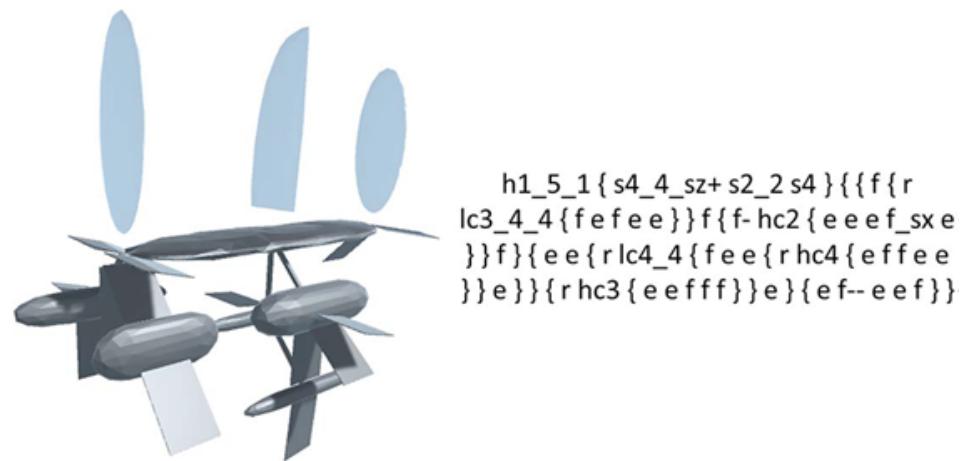


Figure 12. A grammatical string representing a feasible sailboat design [55].

6.1.5. Use of Construction Space for the Design of Additive Production

Through logical and integrated product designs, materials, and production methods, this project's additive manufacturing design maximizes product performance. It is difficult to find design solutions in such a complex design area. It is notable that no current design assistance solution is both quick and tailored to the design process. In this research, a holistic strategy for designing and optimizing the design across the various phases of the design process is provided. In particular, a two-step design process based on different models for the execution and detailed design stages is offered. The Bayesian network classifier is utilized as the reasoning framework for investigating the design space during the design

phase, whilst the Gaussian process regression model is employed as the evaluation function for the optimization approach to utilize the design (construction) space during the detailed design. On the basis of a single dataset generated by the Latin hypercube sampling technique, and improved using the Monte Carlo Markov chain sampling method, these models were developed [88]. This cost-effective data-based method is exemplified in the design of a customizable ankle brace that has mechanical characteristics that can be tuned utilizing a high-expansion design concept with bespoke stiffnesses [89].

6.1.6. Generative Design and Verification of 3D Conceptual Wheel Models

This investigation focuses on an engineering design that combines artificial intelligence (AI) with computer-aided design (CAD) and computer-aided engineering (CAE). In this research, a CAD/CAE framework based on deep learning was developed for the conceptual design phase, which automatically creates 3D CAD drawings and analyzes their technical performances. The following steps comprise the proposed framework: 2D generative design, dimensionality reduction, latent space experiment design, automated production of 3D CAD models, automatic modeling of CAE findings, transfer learning, visualization, and analysis. On the basis of a case study involving the design of a road wheel (see Figure 13), the suggested framework demonstrates that AI may be realistically included into a final product design project [90]. Using this framework, engineers and industrial designers may review a large number of 3D CAD models together with the findings of technical performance (CAE analysis) predicted by AI to determine the ideal conceptual design variations for the future phase of detailed designs [91].

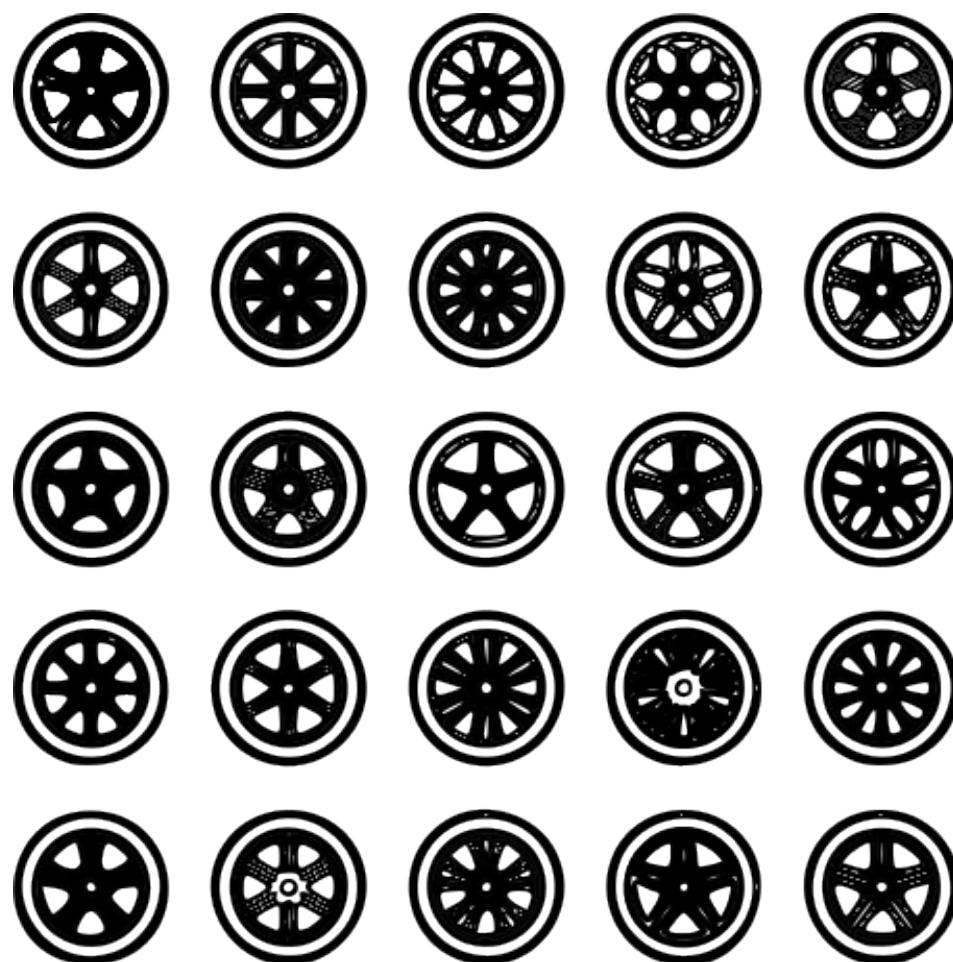


Figure 13. Automatic generation of 2D wheel disk images [92].

6.1.7. Machine Learning-Assisted Propeller Design

In Maritime and subsonic aircraft, propellers are some of the most common propulsion mechanisms utilized to create thrust from the rotating motion of the engine. Due to their simplicity, durability, and great efficiency, propellers have been the most popular design options used in the last century [93]. Nonetheless, it remains difficult to identify the best application-specific shape. This study addresses the use of contemporary and quickly advancing machine learning (ML) methods in the production of new designs. It leverages a massive collection of previously existing parametric design patterns and includes technical knowledge, with extremely realistic simulation models; thus, the design process was approached as a supervised learning problem.

The objective was to create and test machine learning models for parametric propeller designs (see Figure 14) using application-specific restrictions. The outcome was the discovery of an unorthodox propeller geometry with extremely high efficiency. The exceptionally high accuracy of validation data is exemplary, and these results suggest that further progress in this direction can help examine new, unrecognized, effective designs and save designers a great deal of time; further progress can also help to alleviate the challenges of validating these designs beyond accuracy [94].

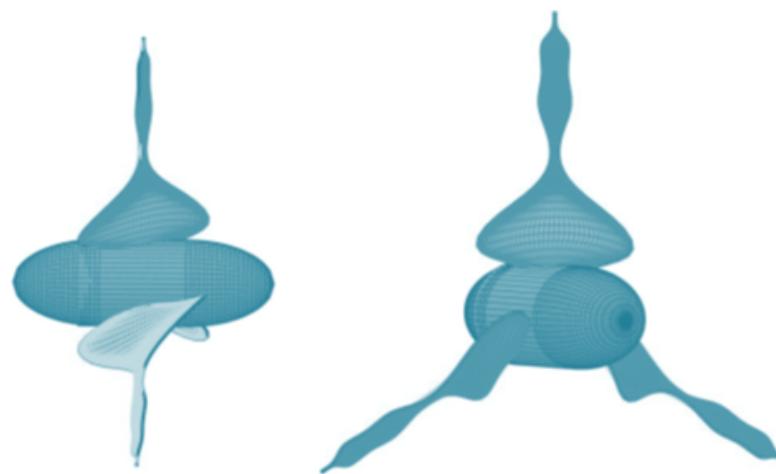


Figure 14. Ship propellers designed using machine learning [94].

6.1.8. Evaluation of Steel Structures Using Machine Learning

In this study, a novel design method based on an iterative machine learning algorithm was presented to expedite the topological investigation of shell structures (see Figure 15) with planar surface compression, while taking structural performances and design restrictions into consideration. Neural networking enables the training of a surrogate model to expedite the assessment of the structural performance of a variety of potential structural forms without the need for a substantially slower process of locating geometric shapes [95]. As the principal tool of structural designs, methods of identifying geometric shapes of 3D graphic statics are utilized to produce a single-layer shell by compressing flat surfaces. Under identical boundary conditions, the partitioning of the force diagram and its polyhedral cells using different rules results in topologically distinct structures that possess varying carrying capabilities. The solution space for all potential forms with pure compression under a given boundary condition is enormous, making it very hard to iterate over all forms in order to locate optimal solutions. After training with iterative active sampling, the surrogate model can analyze input data, including distribution rules, and forecast the structural performance and design restrictions of planar surfaces in milliseconds. Consequently, it is feasible to analyze the nonlinear interactions between all distribution rules and the chosen structural performance metrics, and then display the full solution space. As a consequence, a number of design solutions with specific assessment criteria have been discovered, demonstrating the effectiveness of this form-finding technique. Moreover,

given the overall training time of the neural network model, the suggested framework is still quicker than a conventional optimization technique, such as a genetic algorithm that can only discover optimum values. This procedure yields interactive sampling approaches in which machine learning models aid the designer in picking and controlling various design strategies by giving real-time feedback on the impacts of chosen parameters on design results [96].

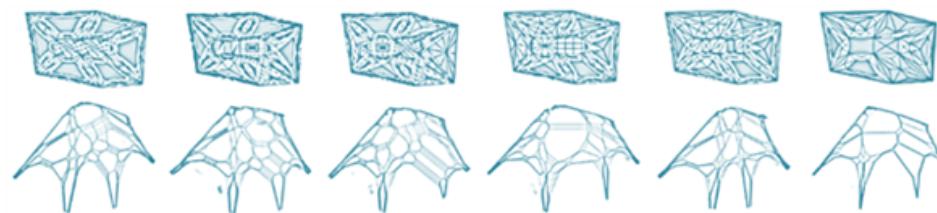


Figure 15. Examples of asymmetric structural shapes [96].

6.1.9. Visual Search Application for Machine Components

This study examined a comprehensive, annotated benchmark of mechanical components for classification and retrieval operations. It is a comprehensive data collection of 3D mechanical component objects (see Figure 16). This dataset enables the data-driven study of mechanical component symptoms. Examining the form descriptor of mechanical components is vital for computer vision and industrial applications. However, developing annotated datasets on mechanical components has not received much attention. Obtaining 3D models is difficult because annotating mechanical components requires technical expertise. The primary contributions of this study are the establishment of a large reference collection of annotated mechanical components, the definition of a hierarchical taxonomy for mechanical components, and a comparison of the performances of deep learning classifiers for shape analysis applied to mechanical components. Seven cutting-edge techniques for categorizing deep learning into three categories, namely cloud points, volume representation in voxel grids, and representation-based representation, were examined using an annotated dataset. Autodesk, the developer of Design Graph software, is conducting similar research [97,98].



Figure 16. Machine parts database [99].

6.2. Optimization of Strength Designs

Computational optimization methods have matured in recent years due to extensive research conducted by applied mathematicians and engineers. These practices apply to many of the day-to-day applications we use as humans. Even in this context, the utilization and principles of artificial intelligence, specifically machine learning, are prevalent.

The following subchapters present studies and research that address the issue of optimization in the design of various mechanical systems.

6.2.1. Machine Learning as a Substitute for the Finite Element Method

Current maintenance intervals for mechanical systems in transportation are based on system life, which results in expensive maintenance planning and often puts passengers at risk. In the future, the actual usage of the vehicle will be utilized to anticipate the stresses in its design, therefore enabling the development of a particular maintenance plan. Machine learning (ML) methods may be used to transfer a subset of real-time design measurements to a finite element analysis-based, high-fidelity model of the same system (FEM). Consequently, the ML technique based on finite elements directly calculates the stress distribution across the system during operation, hence enhancing the capacity to create safe and effective maintenance procedures [100]. In this research, an alternative finite element method based on machine learning methods was presented for estimating the time-varying response of a one-dimensional beam. Several regression models, such as decision trees and artificial neural networks, were created; their effectiveness in estimating the stress distribution in the beam structure was evaluated. Substitute finite element models based on ML algorithms may predict the beam response correctly; however, artificial neural networks have produced more precise results [101].

6.2.2. Application of Machine Learning for the Calculation of Pressure Equipment

The use of finite element analysis (FEA) is widely prevalent in the design of pressure equipment. While the finite element approach offers stress distribution in the geometry of pressure equipment, the analyst must manually specify specific parameters and boundary conditions in order to conduct the analysis. The purpose of this research was to enhance or replace manual analytical processes using machine learning. Two distinct models have been created to replace two similar manual pressure equipment analysis processes. The models were trained using 605 distinct datasets derived from the stress study from a common region of discontinuities in pressure equipment. In order to detect discontinuities and anticipate linearized stresses, machine learning models were created, thereby accelerating the analytical process. The findings indicate that the trained models are sufficiently accurate at greatly augmenting the analytical procedure [102].

6.2.3. Optimization of Mechanical Micro-Mixer Design Using Machine Learning

Due to the expenses involved with producing and testing multiple prototypes, it is often challenging to establish an ideal multipurpose design in practical engineering applications. Therefore, it is advised to use calculation tools. In this study, machine learning approaches (solved as design optimization with machine learning help) were utilized to build a mechanical mixing device consisting of a microchannel-enclosed rectangular column. The random forest classifier was trained to predict which geometries might result in vortex scattering. Next, a multi-objective optimization issue was explored, which consisted of decreasing the needed pumping power and optimizing the mixing efficiency within the restrictions of a given design. If additional trained data were necessary for alternative configurations, the random forest classifier could have been used to predict whether or not it was worthwhile to simulate a new configuration, thereby preventing the simulation of unnecessary computationally intensive cases and accelerating the data-driven process. The performance of the optimum designs derived from the application of the genetic algorithm on surrogate samples was simulated and shown. Even under very unfavorable mixing circumstances and a somewhat low Reynolds number, optimal geometric arrangements give optimal mixing efficiencies at very low short-channel pumping costs, exceeding conventional technology. By using form parameterization methodologies, the MLADO framework that guides this research may be expanded and automated for other comparable engineering design processes of any size [103].

6.2.4. Use of Machine Learning for the Prediction and Optimization of Mechanical Systems

This study examined the precision of predictions made by multiparametric models obtained from numerical data. There were three distinct mechanical test scenarios utilized

to produce the data. Models were constructed from the data to predict changes in attributes in response to arbitrary changes in input parameters. Various modeling methodologies were assessed based on the accuracy of their predictions. By providing machine learning toolboxes, polynomial matrix equations were contrasted with regression models and neural network models. Model similarities and differences were discussed. A matrix-based exponential model was developed to improve the precision of certain applications. The effects and causes of the model prediction accuracies were evaluated. This defines the basic conditions for developing useful models. This resulted in a comparison of modeling methodologies in terms of their physical plausibility and efficacy. There is a correlation between data production and the training process and efficiency. For one of the example situations (see Figure 17), a prediction model was used to show its application and capabilities. Using the model equation, the value of the sanction function in an optimization job with various inputs and outputs was calculated. By updating the material parameters, the outcome of the optimization involved the adjustment of the four natural frequencies to the observed values. In every other instance, sensitivity analyses were conducted, including the validation of numerical findings [104].

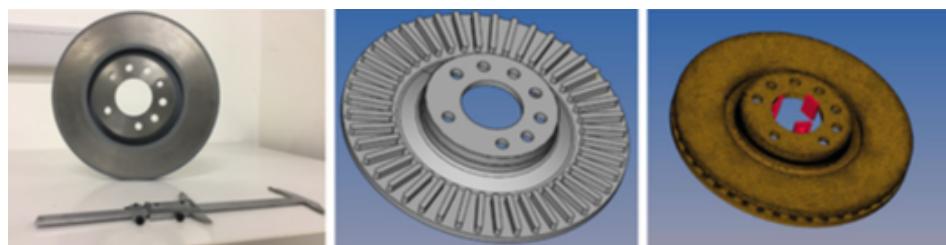


Figure 17. Sample brake disc geometry [104].

6.2.5. Topology Optimization

Recent advancements in design optimization have the potential to significantly enhance the performances of mechanical components and systems. In combination with additive manufacturing, topology optimization is one of the numerical approaches used to algorithmically build optimized designs that influence the mechanical design of existing hardware. Sadly, many of these algorithms may require substantial human configuration and control, particularly the debugging parameters that govern the algorithm's performance and convergence. This research proposes a machine learning-based framework to recommend tuning parameters to users, therefore avoiding the expensive trial-and-error process of human tuning. The program collects the debugging parameters from the repository of past comparable issues, which are assessed using different parameters, depending on the problem information, and refines them using the Bayesian optimization technique for the present problem. The technique is illustrated on a basic topological optimization problem with the objective of achieving a decent topological optimization solution and then determining the appropriate "compromise" between the solution quality and necessary computing time. The objective is to decrease the number of "unnecessary" debugging starts needed for manual debugging alone. With more improvement, this system may eventually be applicable to enterprise-level analysis and optimization problems. Topology optimization is one example, but the framework may also be used for other optimization issues, such as form and dimensioning in highly accurate physics-based analytical models [105].

6.2.6. Design of Hollow Section Columns Using Machine Learning Methods

Many current techniques for designing the load-bearing capacity of round, hollow, stainless-steel columns have been established for a particular class, taking the global deflection failure approach into consideration. However, various grades of stainless steel have significantly varying material characteristics, and hollow section columns are susceptible to local deflection, global deflection, and both global and local interactive deflections. In this research, a framework based on machine learning was employed to build a standard de-

sign process applicable to various grades of stainless steel and failure mechanisms. First, 39 experiments on cold-formed hollow section columns were conducted (see Figure 18). Material characteristics, flaws, load and deformation curves, and failure mechanisms were described in detail [106]. Then, test data for these stainless-steel columns were gathered, and a database of 280 columns was generated. Then, two machine learning methods, random forest and extreme gradient boosting, were used to forecast column load capacities based on four different kinds of input characteristics. Using all design inputs, the random forest algorithm had maximum prediction accuracy. The inclusion of the yield strength to Young's modulus ratio as an input parameter considerably enhanced the accuracy of the random forest technique when based on complicated factors (i.e., the dimensionless cross-sectional area and bar slenderness) [107].

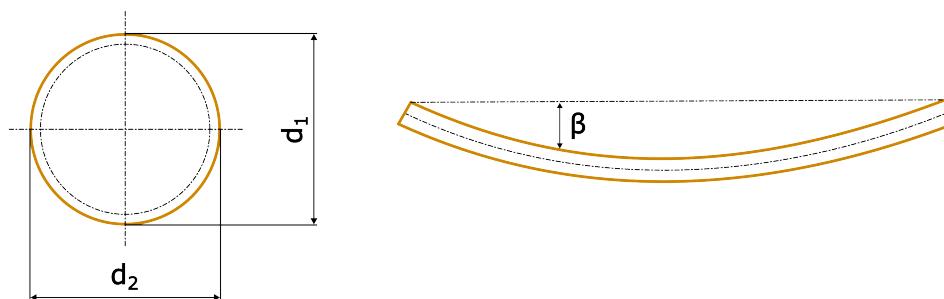


Figure 18. Measurement of imperfections on columns [107].

6.2.7. Prediction of Load-Bearing Capacity of Double-Cut Screw Joints

This research involves the ground-breaking use of machine learning to forecast the load-bearing capabilities of double-cut structural steel bolted joints. A database containing 443 experimental datasets with input characteristics, such as standardized end distances, edge distances, bolt spacing, transverse to load directions, sheet steel strength ratios, load cell rows, joint configuration, and standard load capacity, was compiled for the first time. For this application, eleven machine learning approaches were studied. The investigation of the significance of the factors revealed that the normalized distances between the ends and edges had the largest influence on the final strength. Using different statistical indicators, the performances of the models were assessed and compared to current formulations and requirements of the proposed legislation. The model with the greatest coefficient of determination, the lowest mean absolute error, and the lowest mean square error was the random forest model. In contrast to previous models, which are particular to individual steel grades and give separate equations for various failure modes, machine learning models have achieved accurate, integrated, and generic approaches to prediction. Intriguingly, these models demonstrate that the ratio of steel tensile strength to compressive strength and the number of screw rows, which are presently disregarded in the design recommendations, considerably influence bearing strength (by about 10%). A user-friendly interface comprising all of the planned machine learning algorithms was created, which helped the design of double-cut screw joints and acted as a teaching and research resource [108].

6.2.8. Design and Optimization of Pneumatic Gear Drive

Electro-pneumatic drives play a vital role in a variety of sectors, including in the construction of large vehicles. This study investigates the topic of operating an automated manual gearbox whose actuator is a double-acting cylinder with a floating piston and a reserved internal position. When constructing the control of electro-pneumatic cylinders, it is important to execute control with a fixed value on a nonlinear system when the airflow is provided by disproportionate valves, as is the case here. Due to the fact that both the system model and the qualitative management objectives may be expressed as partly observable Markov decision-making processes, machine learning frameworks are obvious options for addressing such management issues. This research analyzes six distinct options for this purpose. In a simulated environment, the performances and strategic choices of these six

approaches were compared. To illustrate the applicability of the idea, validation tests were conducted on an actual transmission system and implemented on the vehicle's control unit. In this instance, the policy gradient agent was chosen based on implementation limitations and computing capability. The findings indicate that the given approaches are appropriate for the control of floating piston cylinders and may be applied to other mechanical fluid drives and nonlinear control problems with fixed values [109].

6.3. Getting Human Preferences and Design Strategies

Recent advances in artificial intelligence offer the perfect opportunity to integrate this technology (including machine learning) into the human design and construction teams. To date, machine learning approaches have not paid enough attention to the consideration of human aspects, which play an important role in the design process. These subchapters present research and studies in the field of obtaining human preferences and strategies, which can be used in decision-making in the design and development of various products.

6.3.1. Data-Driven Methodology for Creating Customer Selection Sets Using Online Data and Customer Reviews

Recent innovations in technological design include a selection model that takes client preferences into account. When constructing a choice model that enables one to gain an accurate estimate of the parameters associated with product features, a solid collection of options is crucial. However, the selected file information is often unavailable [110]. This research offers a mechanism for constructing customer samples using web data and customer evaluations in the absence of real samples or sociodemographic consumer data. The approach consists of three primary components, i.e., categorizing items according to their qualities, grouping customers according to their evaluations, and building sample files based on a selection probability scenario based on product and customer groups. The normalized scenario describes the suggested scenario, which multiplies the proportions of product clusters and consumers to generate a probabilistic distribution of choice. A linear combination of product qualities alone and a feature that incorporates the interplay of product attributes and customer reviews are provided as utility functions. The approach was applied to a collection of laptop data. Predicting the test file data, the normalized scenario obtains much better outcomes than the baseline (random) scenario. In addition, the addition of customer evaluations to the utility function considerably improves the model's predictive ability. Using product attribute data and customer evaluations to produce samples yields sample models with more predictive ability than randomly generated samples, according to research [111].

6.3.2. Obtaining Customer Insights on Product Sustainability from Online Reviews

For a product to be successful on the market, its designers must produce items that are not only real but also sustainable in terms of consumer perceptions; both the reality and perceptions of competition may vary significantly. This study details the design technique used for determining the perception of sustainable features via the gathering of online reviews, their human annotations via crowdsourcing, and the processing of review fragments annotated by a natural language machine learning algorithm. As indicated by the observed discrepancy between the perception of sustainability and the actuality of sustainable design, the research demonstrates that a number of critical environmental sustainability problems, such as energy and water use, did not have a substantial influence on consumer sentiment. Based on these outcomes, internet reviews may help designers acquire competitive traits for future design studies and produce products that are compatible with sustainable consumer values [112].

6.3.3. A Data-Driven Approach to Identify the Context of Product Use from Online Customer Reviews

This study presents a data-driven technique for identifying product use scenarios automatically from online customer evaluations. The environment of product usage influences

product design, user behavior, and consumer happiness. Prior research identified use settings through a survey-based technique or by subjective means. In contrast, the suggested technique employs natural language machine learning and processing methods to discover and aggregate use contexts from a large number of customer evaluations. In addition, aspect analysis is utilized to extract sentiment within a specific phrase context. The outcome demonstrates that the approach is capable of capturing important product usage contexts and clustering bigrams associated with comparable use contexts. Aspect sentiment analysis enables the observation of a product's position relative to its rivals in a certain context of usage. This insight may highlight to the product creator a need to enhance the product. It may also signal a possible marketing opportunity in an application scenario where the majority of existing items are adversely evaluated by consumers. Due to a minor linear connection for the majority of use contexts in the case study, it suggests that the overall evaluation may not be a significant predictor of the customer's mood in regard to a specific use scenario [113].

6.3.4. Mining and Representing the Conceptual Space of Existing Ideas for the Purpose of Targeted Idea Generation

Frequently, design innovation initiatives produce a great number of design ideas from designers, consumers, and, increasingly, regular internet users. Such data are often utilized for selection and execution, but they may also serve as sources of inspiration for the generation of further ideas. Specifically, the basic concepts underlying the original ideas may be recombined to form new concepts. Obtaining concepts from raw lists of ideas and data sources in order to inspire or develop new ideas is not a simple operation. The fact that data on concepts is often presented in unstructured natural languages is a key issue. In this study, a methodology is developed that employs natural language processing to extract keywords as elementary concepts embedded in massive descriptions of ideas and represents the space of elementary concepts in a core-periphery structure to facilitate the recombination of elementary concepts into new ideas. The methodology is applied to the extraction and representation of the conceptual space that underlies the massive ideas gained from crowdsourcing and is used to generate new ideas for future transport system designs in a real project funded by the public sector through the use of people and automated computer programs. The investigation into the processes and outcomes of human and machine recombination provides information on the future directions of artificial intelligence research in the domain of design concepts [114].

6.3.5. Transferring Design Strategies from Human to Computer

Solving every design issue requires planning and developing techniques that identify and prioritize subprocesses. It is a talent that designers acquire through time and then use to tackle comparable situations. This transfer of design techniques has not, however, been successfully modeled or implemented inside computational agents. This paper proposes a probabilistic model-based method for representing design methods. The model offers a method for generating new designs based on specific design strategies as the configuration design challenge is gradually addressed. This experiment also demonstrates that this probabilistic representation may be used to transmit human designer techniques to computer design agents in a generic and effective manner. This transfer-based method enables the identification of high-performance designer behavior and its application to the management of computational design agents. Lastly, the agents exemplify the core behavior of transfer learning since the transfer of design techniques across challenges has resulted in enhanced agent performance. This research employs the CISAT (cognitively inspired simulated annealing teams) framework, an agent-based model that replicates human problem-solving in configuration design challenges [115].

6.3.6. Mimicking Human Designers Through Deep Learning

People, as designers, have rather diverse problem-solving techniques. Computer agents, on the other hand, have access to substantial computational resources for solving specific design issues. Consequently, if agents can learn from human behavior, a synergistic team may be established to handle challenges involving humans and artificial intelligence. This study proposes a method for extracting human design strategies and implicit norms from historical human data and using them to create designs. This involves the creation and implementation of a two-tiered framework that mimics human design techniques via observational learning. This system uses deep learning components to build designs without explicit goals and performance parameter information. The framework is meant to engage with the issue using a visual interface, similar to how individuals solve the challenge. It is taught to imitate a group of human designers by monitoring the sequences of their design states, without introducing a problem-specific modeling interest or extra issue knowledge. In addition, a final agent is constructed that employs this deep learning architecture as its foundation, together with image processing techniques to map pixel motions to the design as a method for design development. Finally, the designs created by the calculation teams of these agents are compared with real human data for the teams tasked with addressing roof truss designs. (see Figure 19) The findings demonstrate that these agents are capable of independently designing viable and effective roof truss structures, indicating that this technique enables agents to acquire useful design methodologies [116].

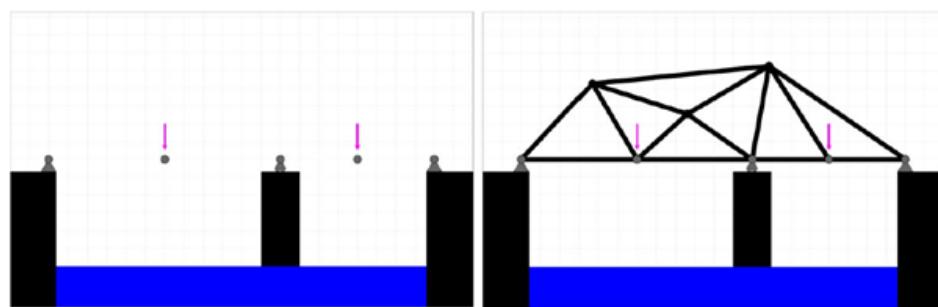


Figure 19. Display of the truss design software interface with parameters and possible operations [116].

7. Discussion

Neural networks have been widely used in mechanical engineering and product designs for various purposes such as simulation, optimization, prediction, control, and design strength optimization of various structures such as bridges, buildings, and vehicles. They also have been used in engineering design to obtain human preferences and design strategies by analyzing large datasets of design choices, user feedback, and expert knowledge. Here are some common types of neural networks used in mechanical engineering:

7.1. Convolutional Neural Networks (CNNs)

CNNs are commonly used in mechanical engineering for image recognition, analysis, and classification. In product designs, they can be used for visual inspections, quality control, and defect detection in manufacturing processes. Convolutional neural networks can be used for the image-based design optimization of structural components, such as beams, columns, and trusses. They can analyze the 2D or 3D images of components to identify design features and optimize the strength and weight of the component. CNNs are also used for image analysis and recognition. In designs, they can be used to analyze images of existing products, user interfaces, or design features, and identify key design features that are preferred by users. These preferences can then be used to guide the design of new products.

7.2. Recurrent Neural Networks (RNNs)

RNNs are used in mechanical engineering for time-series data analysis, prediction, and control. In product designs, they can be used for predictive maintenance, anomaly detection, and optimization of manufacturing processes. Recurrent neural networks can be used to optimize the strength designs of time-varying structures, such as wind turbines, offshore platforms, and bridges. They can analyze the time-series data of loads and stresses and predict the optimal design for a given set of inputs. RNNs can be also used for natural language processing and sequence modeling. In designs, they can be applied to analyze user feedback, reviews, or comments, to identify key preferences and design strategies. They can also be used to generate natural language descriptions of design features and use these descriptions to guide the design process.

7.3. Multilayer Perceptrons (MLPs)

MLPs are used in mechanical engineering for pattern recognition, classification, and regression analysis. In product designs, they can be used for material selection, design optimization, and performance prediction. MLPs are commonly used for optimizing strength designs in mechanical engineering. They can be trained on a dataset of material properties, geometries, and loads, and used to predict the optimal design for a given set of inputs. MLPs can also be used for multi-objective optimization, where multiple design objectives, such as weight, cost, and strength, are optimized simultaneously.

7.4. Autoencoders

Autoencoders are used in mechanical engineering for dimensionality reduction, feature extraction, and anomaly detection. In product design, they can be used for design space exploration, optimization, and fault detection. Autoencoders can be used to optimize strength designs by reducing the dimensionality of the input data. They can extract key features from a large dataset of designs and use those features to optimize new designs. Autoencoders can also be used for generative designs, where new designs are generated based on the learned features. Autoencoders can be used for feature extraction and the dimensionality reduction of design data. They can be used to identify the most important design features and relationships between them. This can help designers in understanding which design features are preferred by users and which features are less important.

7.5. Deep Belief Networks (DBNs)

DBNs are used in mechanical engineering for feature extraction, classification, and prediction. In product design, they can be used for performance prediction, design optimization, and fault diagnosis.

7.6. Generative Adversarial Networks (GANs)

GANs are used in mechanical engineering for tasks such as generating new design concepts, simulations, and optimizations. In product designs, they can be used for design space exploration, optimization, and generating new product designs. GANs can be used for generating new design concepts and exploring design spaces. They can be trained on large datasets of design features, preferences, and user feedback to generate new design concepts that meet user preferences. GANs can also be used to create variations of existing designs and explore new design possibilities.

7.7. Deep Reinforcement Learning (DRL)

DRL can be used to optimize strength designs by learning from trial and error. It involves training a neural network to take actions that maximize a reward signal in each environment. In strength design, DRL can be used to explore the design space and find the optimal design for a given set of objectives. It is also possible to use DRL for learning design strategies and decision-making processes. It involves training a neural network to take actions that maximize a reward signal in a given environment. In designs, DRL can be

used to learn from user feedback and expert knowledge to develop design strategies that optimize user preferences.

Overall, the use of neural networks in mechanical engineering and product designs is growing rapidly, and their potential applications are vast. Artificial neural networks are still an expanding area of research. However, the selection of a particular type of neural network depends on the specific problem being addressed and the data available for analysis.

8. Conclusions

Modern machine learning (ML) and deep learning (DL) approaches are revolutionizing industries, ranging from transportation to healthcare. These approaches identify patterns in data, develop autonomous systems that exhibit human-like skills, and assist in human decision-making. Deep neural networks, and other current ML methods, are enabling significant advances in AI. ML approaches are being increasingly used by researchers in mechanical design and optimization to help solve tasks ranging from preference modeling to uncertainty quantification in high-dimensional design optimization problems. This survey lists important advances in these areas. In particular, the overall goal of these applications and practices is to rapidly reduce the development times for companies and firms so their designers and engineers could focus on other areas of development and mechanical design.

First, we analyzed artificial intelligence, its distribution, advantages, disadvantages, and applications. We discussed machine learning and then deep learning. We presented their procedures, the most common methods, and their distributions. We also compared machine learning and deep learning. In this study, we described how neural networks work and what they are used for. In the next part of the review, we comprehensively discussed the uses and functions of the most commonly used programming languages, frameworks, and software that are used in the field, mostly just in mechanical engineering. We also presented a breakdown of different types of commercial software according to their programming languages. Then, in the following chapters, we described the different data formats that represent inputs to neural networks in the field of mechanical design and optimization. We also presented the most commonly used datasets of these input data. In the second half of the review, we discussed the use of machine learning and deep learning in machine design and optimization. We presented individual examples that were studied in recent years by researchers from all over the world. We have divided this section into several areas: product design, strength optimization, and finally, the area of human preference elicitation and strategies in engineering design.

In the discussion section, we presented opportunities for further research and advancement of machine learning and deep learning in mechanical design and optimization using various types of neural networks.

As artificial intelligence continues to expand in today's world, it is expected that future mechanical designers will encounter machine learning in their daily lives. This means that educational activities will have to adapt to this trend.

Author Contributions: Conceptualization, J.J., J.O. and M.C.; methodology, J.J. and J.O.; software, J.J. and J.O.; validation, J.J., J.O. and F.B.; formal analysis, J.J. and M.C.; investigation, S.H. and J.O.; resources, J.J.; data curation, J.O.; writing—original draft preparation, J.J. and E.S.; writing—review and editing, F.B. and J.O.; visualization, J.J. and J.O.; supervision, F.B.; project administration, F.B. and J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovak Research and Development Agency under grant number APVV-18-0450. The research focused on investigating the influence of design parameters of special transmissions with a high gear ratio in relation to their kinematic properties.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: This study was supported by the Slovak Research and Development Agency under contract no. APVV-18-0450. The research focused on investigating the influence of design parameters of special transmissions with a high gear ratio in relation to their kinematic properties.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. D'Angelo, G.; Castiglione, A.; Palmieri, F. A cluster-based multidimensional approach for detecting attacks on connected vehicles. *IEEE Internet Things J.* **2020**, *8*, 12518–12527. [[CrossRef](#)]
2. D'Angelo, G.; Palmieri, F. Discovering genomic patterns in SARS-CoV-2 variants. *Int. J. Intell. Syst.* **2020**, *11*, 1680–1698. [[CrossRef](#)]
3. Lei, Y.; Yang, B.; Jiang, X.; Jia, F.; Li, N.; Nandi, A.K. Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mech. Syst. Signal Process.* **2020**, *138*, 106587. [[CrossRef](#)]
4. Artificial Intelligence. Available online: <https://www.javatpoint.com/artificial-intelligence-ai> (accessed on 2 February 2023).
5. Advantages and Disadvantages of Artificial Intelligence. Available online: <https://towardsdatascience.com/advantages-and-disadvantages-of-artificial-intelligence-182a5ef6588c> (accessed on 2 February 2023).
6. What Is Machine Learning? Available online: <https://www.ibm.com/topics/machine-learning> (accessed on 2 February 2023).
7. Reinforcement Learning. Available online: <https://www.geeksforgeeks.org/what-is-reinforcement-learning/> (accessed on 2 February 2023).
8. What Is Deeplearning and How Does It Work. Available online: <https://towardsdatascience.com/what-is-deep-learning-and-how-does-it-work-2ce44bb692ac> (accessed on 3 February 2023).
9. Upgrad. Available online: <https://www.upgrad.com/blog/biological-neural-network/> (accessed on 22 May 2023).
10. Neural Network: Architecture, Components & Top Algorithms. Available online: <https://www.upgrad.com/blog/neural-network-architecture-components-algorithms/> (accessed on 3 February 2023).
11. Introduction to Learning Rules in Neural Network. Available online: <https://data-flair.training/blogs/learning-rules-in-neural-network/> (accessed on 4 February 2023).
12. Machine vs. Deep Learning as Artificial Intelligence Principle Outline Diagram. Available online: https://www.123rf.com/photo_178844732_stock-vector-machine-vs-deep-learning-as-artificial-intelligence-principle-outline-diagram.html (accessed on 4 February 2023).
13. AI Technical Machine vs. Deep Learning. Available online: <https://lawtomated.com/a-i-technical-machine-vs-deep-learning/> (accessed on 4 February 2023).
14. Goldsborough, P. A tour of tensorflow. *arXiv* **2016**, arXiv:1610.01178.
15. What Is Python? Executive Summary. Available online: <https://www.python.org/doc/essays/blurb/> (accessed on 29 June 2021).
16. Business 2 Community. Available online: <https://www.business2community.com/big-data/python-ai-why-python-is-better-for-machine-learning-and-ai-02389380> (accessed on 29 June 2021).
17. Simpli Learn. Available online: <https://www.simplilearn.com/what-is-cpp-programming-article> (accessed on 28 March 2023).
18. Geeks for Geeks. Available online: <https://www.geeksforgeeks.org/introduction-to-machine-learning-in-r/> (accessed on 28 March 2023).
19. Turing. Available online: <https://www.turing.com/kb/scope-of-machine-learning-using-java-and-nlp#fast-execution> (accessed on 28 March 2023).
20. What Is A Machine Learning Framework & 10 That You Need To Know. Available online: <https://analyticsindiamag.com/machine-learning-framework-10-need-know/> (accessed on 5 February 2023).
21. Top 8 Deep Learning Frameworks. Available online: <https://marutitech.com/top-8-deep-learning-frameworks/> (accessed on 5 February 2023).
22. ML Frameworks Compared: scikit-learn, Tensorflow, PyTorch and More [Updated]. Available online: <https://www.netguru.com/blog/top-machine-learning-frameworks-compared> (accessed on 5 February 2023).
23. Apache Mahout: Machine Learning on Distributed Dataflow Systems. Available online: <https://www.jmlr.org/papers/v21/18-800.html> (accessed on 5 February 2023).
24. Top 10 Machine Learning Frameworks. Available online: <https://www.promptcloud.com/blog/top-10-machine-learning-frameworks/> (accessed on 5 February 2023).
25. Top 15 Frameworks for Machine Learning Experts. Available online: <https://www.kdnuggets.com/2016/04/top-15-frameworks-machine-learning-experts.html> (accessed on 5 February 2023).
26. Best Machine Learning Software. Available online: <https://project-management.com/best-machine-learning-software/> (accessed on 5 February 2023).
27. Is Octave Good for Machine Learning? Available online: <https://datasciencenerd.com/is-octave-good-for-machine-learning/> (accessed on 5 February 2023).
28. What Is MATLAB? Available online: <https://cimss.ssec.wisc.edu/wxwise/class-aos340/spr00/whatisMATLAB.htm> (accessed on 5 February 2023).
29. Maple (Software). Available online: [https://en.wikipedia.org/wiki/Maple_\(software\)](https://en.wikipedia.org/wiki/Maple_(software)) (accessed on 5 February 2023).

30. Wolfram Mathematica. Available online: <https://www.wolfram.com/mathematica/> (accessed on 5 February 2023).
31. Ansys. Available online: <https://www.ansys.com/technology-trends/artificial-intelligence-machine-learning-deep-learning> (accessed on 12 February 2023).
32. Esi-Group. Available online: <https://www.esi-group.com/resources/technical-paper/integration-machine-learning-and-visualization-effective-design-exploration> (accessed on 12 February 2023).
33. Altair. Available online: <https://www.altair.com/designai/> (accessed on 12 February 2023).
34. Materialise. Available online: <https://www.materialise.com/en/inspiration/articles/machine-learning-layer-image-analysis> (accessed on 12 February 2023).
35. Hexagon. Available online: <https://hexagon.com/products/odyssee> (accessed on 12 February 2023).
36. Numeca. Available online: <https://www.numeca.com/studying-the-nature-of-turbulence-with-artificial-intelligence-and-machine-learning> (accessed on 12 February 2023).
37. Dassault Systemes. Available online: <https://www.3ds.com/products/analytics-big-data-artificial-intelligence> (accessed on 12 February 2023).
38. Blog.rhino3d. Available online: <https://blog.rhino3d.com/2017/08/machine-learning-for-rhino-and.html> (accessed on 12 February 2023).
39. PTC. Available online: <https://www.ptc.com/en/technologies/iiot/ai-machine-learning> (accessed on 12 February 2023).
40. Comsol. Available online: <https://www.comsol.com/video/keynote-predicting-corrosion-with-machine-learning-and-simulation> (accessed on 12 February 2023).
41. Autodesk. Available online: <https://adsknews.autodesk.com/en/views/ai-lab-cvpr-2022/> (accessed on 12 February 2023).
42. Ideastatica. Available online: <https://www.ideastatica.com/support-center/connection-browser-properties-parameters-and-filtering> (accessed on 12 February 2023).
43. Siemens. Available online: <https://newsroom.sw.siemens.com/en-US/siemens-nx-additive-optimization-machine-learning/> (accessed on 12 February 2023).
44. Noesis. Available online: <https://www.Noesissolutions.com/about-Noesis-solutions/news-events/optimus-rev-2018-1-introduces-new-modeling-methods-boosted-by-machine-learning> (accessed on 12 February 2023).
45. Aveva. Available online: <https://www.aveva.com/en/solutions/digital-transformation/artificial-intelligence/> (accessed on 12 February 2023).
46. Beta-Cae. Available online: https://www.beta-cae.com/ml_toolkit.htm (accessed on 12 February 2023).
47. Daneshmand, M.; Helmi, A.; Avots, E.; Noroozi, F.; Alisinanoglu, F.; Arslan, H.S.; Anbarjafari, G. 3d scanning: A comprehensive survey. *arXiv* **2018**, arXiv:1801.08863.
48. Remondino, F. From point cloud to surface: The modeling and visualization problem. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2003**, *34*, 12.
49. Ranjan, A.; Bolkart, T.; Sanyal, S.; Black, M.J. Generating 3D faces using convolutional mesh autoencoders. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 704–720.
50. Cheng, S.; Bronstein, M.; Zhou, Y.; Kotsia, I.; Pantic, M.; Zafeiriou, S. Meshgan: Non-linear 3d morphable models of faces. *arXiv* **2019**, arXiv:1903.10384.
51. Zhang, Z.; Wang, Y.; Jimack, P.K.; Wang, H. MeshingNet: A new mesh generation method based on deep learning. In Proceedings of the Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, 3–5 June 2020; Part III 20; pp. 186–198.
52. Wang, L.; Chan, Y.C.; Ahmed, F.; Liu, Z.; Zhu, P.; Chen, W. Deep generative modeling for mechanistic-based learning and design of metamaterial systems. *Comput. Methods Appl. Mech. Eng.* **2020**, *372*, 113–377. [CrossRef]
53. Chen, W.; Fuge, M. Synthesizing designs with interpart dependencies using hierarchical generative adversarial networks. *J. Mech. Des.* **2019**, *141*, 111403. [CrossRef]
54. Chakrabarti, A.; Shea, K.; Stone, R.; Cagan, J.; Campbell, M.; Hernandez, N.V.; Wood, K.L. Computer-based design synthesis research: An overview. *J. Comput. Inf. Eng.* **2011**, *2*, 021003. [CrossRef]
55. Stump, G.M.; Miller, S.W.; Yukish, M.A.; Simpson, T.W.; Tucker, C. Spatial grammar-based recurrent neural network for design form and behavior optimization. *J. Mech. Des.* **2019**, *141*, 124501. [CrossRef]
56. Yang, W.; Ding, H.; Zi, B.; Zhang, D. New graph representation for planetary gear trains. *J. Mech. Des.* **2018**, *140*, 012303. [CrossRef]
57. Hsu, C.H.; Lam, K.T. A new graph representation for the automatic kinematic analysis of planetary spur-gear trains. *J. Mech. Des.* **1992**, *114*, 196–200. [CrossRef]
58. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
59. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; pp. 1–738.
60. Hastie, T.; Tibshirani, R.; Friedman, J.H.; Friedman, J.H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: New York, NY, USA, 2009; pp. 1–758.
61. Image-Net. Available online: <https://www.image-net.org/> (accessed on 20 February 2023).
62. Cocodataset. Available online: <https://cocodataset.org/#home> (accessed on 20 February 2023).
63. Cs.Toronto. Available online: <https://www.cs.toronto.edu/~kriz/cifar.html> (accessed on 20 February 2023).
64. Tensorflow. Available online: <https://www.tensorflow.org/datasets/catalog/mnist> (accessed on 20 February 2023).

65. Paperswithcode. Available online: <https://paperswithcode.com/dataset/pascal-voc> (accessed on 20 February 2023).
66. Paperswithcode. Available online: <https://paperswithcode.com/dataset/kinetics> (accessed on 20 February 2023).
67. Crcv.ucf. Available online: <https://www.crcv.ucf.edu/data/UCF101.php> (accessed on 20 February 2023).
68. Shapenet. Available online: <https://shapenet.org/> (accessed on 20 February 2023).
69. Modelnet.cs.Princeton. Available online: <https://modelnet.cs.princeton.edu/> (accessed on 20 February 2023).
70. Re3data. Available online: <https://www.re3data.org/repository/r3d100012216> (accessed on 20 February 2023).
71. Mcmaster. Available online: <https://www.mcmaster.com/> (accessed on 20 February 2023).
72. Traceparts. Available online: <https://www.traceparts.com/sk> (accessed on 20 February 2023).
73. Grabcad. Available online: <https://grabcad.com/> (accessed on 20 February 2023).
74. 3dcontentcentral. Available online: <https://www.3dcontentcentral.com/> (accessed on 20 February 2023).
75. B2b.Partcommunity. Available online: <https://b2b.partcommunity.com/community/> (accessed on 20 February 2023).
76. Ten-Thousand-Models.Appspot. Available online: <https://ten-thousand-models.appspot.com/> (accessed on 20 February 2023).
77. Arxiv. Available online: <https://arxiv.org/abs/1812.06216> (accessed on 20 February 2023).
78. Github. Available online: <https://github.com/AutodeskAILab/Fusion360GalleryDataset> (accessed on 20 February 2023).
79. Archive.ics.uci.edu. Available online: <https://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set> (accessed on 20 February 2023).
80. Kaggle. Available online: <https://www.kaggle.com/datasets/behrad3d/nasa-cmaps> (accessed on 20 February 2023).
81. Kaggle. Available online: <https://www.kaggle.com/datasets/vinayak123tyagi/bearing-dataset> (accessed on 20 February 2023).
82. Garriga, A.G.; Mainini, L.; Ponnusamy, S.S. A machine learning enabled multi-fidelity platform for the integrated design of aircraft systems. *J. Abbr.* **2019**, *12*, 121405. [CrossRef]
83. Gramblíčka, S.; Kohář, R.; Madaj, R. Construction design automatically adjustable mechanism for crane forks. In Proceedings of the 58th International Conference of Machine Design Departments (ICMD 2017), Praha-Suchdol, Czech Republic, 6–8 September 2017; pp. 100–103.
84. Weis, P.; Kučera, L.; Pecháč, P.; Močilan, M. Modal analysis of gearbox housing with applied load. *Procedia Eng.* **2017**, *192*, 953–958. [CrossRef]
85. Cunningham, J.D.; Simpson, T.W.; Tucker, C.S. An investigation of surrogate models for efficient performance-based decoding of 3D point clouds. *J. Mech. Des.* **2019**, *141*, 121401. [CrossRef]
86. Wu, J.; Zhang, C.; Xue, T.; Freeman, B.; Tenenbaum, J. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 9.
87. Kučera, L.; Gajdošík, T. The vibrodiagnostics of gears. In *Modern Methods of Construction Design: Proceedings of ICMD 2013*; Springer International Publishing: Cham, Switzerland, 2014; pp. 113–118.
88. Majchrak, M.; Kohar, R.; Lukac, M.; Skyba, R. The process of creating a computational 3d model of a harmonic transmission. *MM Sci. J* **2020**, *13*, 3926–3931. [CrossRef]
89. Xiong, Y.; Duong, P.L.T.; Wang, D.; Park, S.I.; Ge, Q.; Raghavan, N.; Rosen, D.W. Data-driven design space exploration and exploitation for design for additive manufacturing. *J. Mech. Des.* **2019**, *141*, 101101. [CrossRef]
90. Majchrák, M.; Kohár, R.; Kajan, J.; Skyba, R. 3d meshing methods of ball-rolling bearings. *Transp. Res. Procedia* **2019**, *40*, 784–791. [CrossRef]
91. Yoo, S.; Lee, S.; Kim, S.; Hwang, K.H.; Park, J.H.; Kang, N. Integrating deep learning into CAD/CAE system: Generative design and evaluation of 3D conceptual whee. *Struct. Multidiscip. Optim.* **2021**, *4*, 2725–2747. [CrossRef]
92. Pneurama. Available online: <https://www.pneurama.com/en/need-wheels> (accessed on 10 March 2023).
93. Galbavý, M.; Pitoňák, J.; Kučera, L. Powershift differential transmission with three flows of power for hybrid vehicles. In *Modern Methods of Construction Design: Proceedings of ICMD 2013*; Springer International Publishing: Cham, Switzerland, 2014; pp. 27–33.
94. Vardhan, H.; Volgyesi, P.; Sztipanovits, J. Machine learning assisted propeller design. In Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems, Nashville, TN, USA, 19–21 May 2021; pp. 227–228.
95. Drbúl, M.; Martíkáň, P.; Bronček, J.; Litvaj, I.; Svobodová, J. Analysis of roughness profile on curved surfaces. *MATEC Web Conf.* **2018**, *224*, 01024. [CrossRef]
96. Zheng, H.; Moosavi, V.; Akbarzadeh, M. Machine learning assisted evaluations in structural design and construction. *Autom. Constr.* **2020**, *119*, 103–346. [CrossRef]
97. Kim, S.; Chi, H.G.; Hu, X.; Huang, Q.; Ramani, K. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; pp. 175–191.
98. Exploring Your Design Information with Machine Learning Using the Design Graph. Available online: <https://www.autodesk.com/autodesk-university/class/Exploring-your-Design-Information-Machine-Learning-Using-Design-Graph-2015#video> (accessed on 5 February 2023).
99. Engineering.Purdue. Available online: <https://engineering.purdue.edu/cdesign/wp/a-large-scale-annotated-mechanical-components-benchmark-for-classification-and-retrieval-tasks-with-deep-neural-networks/> (accessed on 11 April 2023).
100. Kucera, L.; Gajdac, I.; Kamas, P. Computing and design of electric vehicles. In *Modern Methods of Construction Design: Proceedings of ICMD 2013*; Springer International Publishing: Cham, Switzerland, 2014; pp. 105–111.

101. Vurtur Badarinath, P.; Chierichetti, M.; Davoudi Kakhki, F. A machine learning approach as a surrogate for a finite element analysis: Status of research and application to one dimensional systems. *Sensors* **2021**, *5*, 1654. [[CrossRef](#)]
102. Shah, K.; Chandnani, R.; Mavinkurve, U.; Raykar, N. Application of Machine Learning for Design-by-Analysis of Pressure Equipment. In Proceedings of the 2019 International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India, 4–5 January 2019; pp. 1–6.
103. Granados-Ortiz, F.J.; Ortega-Casanova, J. Machine learning-aided design optimization of a mechanical micromixer. *J. Phys. Fluids* **2021**, *6*, 063604. [[CrossRef](#)]
104. Groensfelder, T.; Giebel, F.; Geupel, M.; Schneider, D.; Jaeger, R. Application of machine learning procedures for mechanical system modeling: Capabilities and caveats to prediction-accuracy. *Adv. Model. Simul. Eng. Sci.* **2020**, *7*, 26. [[CrossRef](#)]
105. Lynch, M.E.; Sarkar, S.; Maute, K. Machine learning to aid tuning of numerical parameters in topology optimization. *J. Mech. Des.* **2019**, *141*, 114502. [[CrossRef](#)]
106. Jankejeh, P.; Fabian, P.; Broncek, J.; Shalapko, Y. Influence of tempering on mechanical properties of induction bents below 540 C. *Acta Mech. Et Autom.* **2016**, *2*, 81–86. [[CrossRef](#)]
107. Xu, Y.; Zhang, M.; Zheng, B. Design of cold-formed stainless steel circular hollow section columns using machine learning methods. *Structures* **2021**, *33*, 2755–2770. [[CrossRef](#)]
108. Sarothi, S.Z.; Ahmed, K.S.; Khan, N.I.; Ahmed, A.; Nehdi, M.L. Predicting bearing capacity of double shear bolted connections using machine learning. *J. Mech. Des.* **2022**, *251*, 113–497.
109. Bécsi, T.; Szabó, Á.; Kővári, B.; Aradi, S.; Gáspár, P. Reinforcement learning based control design for a floating piston pneumatic gearbox actuator. *IEEE Access* **2020**, *8*, 147295–147312. [[CrossRef](#)]
110. Gramblíčka, S.; Kohář, R.; Majchrák, M.; Vrabec, M. Contact Analysis of Selected Toothing Contact of the Two-Stage Front Gearbox. In *Current Methods of Construction Design: Proceedings of the ICMD 2018*; Springer International Publishing: Cham, Switzerland, 2020; pp. 263–269.
111. Suryadi, D.; Kim, H.M. A data-driven methodology to construct customer choice sets using online data and customer reviews. *J. Mech. Des.* **2019**, *141*, 111103. [[CrossRef](#)]
112. El Dehaibi, N.; Goodman, N.D.; MacDonald, E.F. Extracting customer perceptions of product sustainability from online reviews. *J. Mech. Des.* **2019**, *141*, 121103. [[CrossRef](#)]
113. Suryadi, D.; Kim, H.M. A data-driven approach to product usage context identification from online customer reviews. *J. Mech. Des.* **2019**, *141*, 121104. [[CrossRef](#)]
114. He, Y.; Camburn, B.; Liu, H.; Luo, J.; Yang, M.; Wood, K. Mining and representing the concept space of existing ideas for directed ideation. *J. Mech. Des.* **2019**, *141*, 121101. [[CrossRef](#)]
115. Raina, A.; Cagan, J.; McComb, C. Transferring design strategies from human to computer and across design problems. *J. Mech. Des.* **2019**, *141*, 114501. [[CrossRef](#)]
116. Raina, A.; McComb, C.; Cagan, J. Learning to design from humans: Imitating human designers through deep learning. *J. Mech. Des.* **2019**, *141*, 111102. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.