

SURVEY

Toward Green AI: A Methodological Survey of the Scientific Literature

ENRICO BARBIERATO  **AND ALICE GATTI** 

Department of Mathematics and Physics, Catholic University of the Sacred Heart, 25133 Brescia, Italy

Corresponding author: Enrico Barbierato (enrico.barbierato@unicatt.it)


ABSTRACT The pervasive deployment of Deep Learning models has recently prompted apprehensions regarding their ecological footprint, owing to the exorbitant levels of energy consumption necessitated by the training and inference processes. The term “Red AI” is employed to denote artificial intelligence (AI) models that undergo training using resource-intensive methodologies on very large datasets. This practice can engender substantial energy usage and emissions of carbon, thereby opposing “Green AI.” The latter concept alludes to AI models designed for similar efficiency and reduced environmental impact. This objective is realized through the utilization of smaller datasets, less computationally intensive training techniques, or sustainable energy resources. While Red AI prioritizes accuracy and performance, Green AI emphasizes efficiency and sustainability. Given that both paradigms exhibit advantages and limitations, the debates around the topics have burgeoned in the scientific arena, delving into novel algorithms, hardware innovations, and improved data utilization techniques aimed at mitigating the ecological consequences of intricate applications such as GPT and BERT. Nevertheless, due to the relative novelty of this debate, not much effort has been dedicated yet to contextualizing the essence of Red AI and the prospects of Green AI in a coherent framework. Within this context, the present work contributes by meticulously delineating both domains through a multifaceted analysis of their causes and ramifications, described from the points of computer architectures, data structures, and algorithms. Additionally, the study reviews notable instances of study cases based on complex Red AI models. The primary contribution of this article encompasses a comprehensive survey of Red and Green AI, stemming from a selection of the literature performed by the authors, subsequently organized into distinct clusters. These clusters encompass i) articles that qualitatively or quantitatively address the issue of Red AI, identifying Green AI as a plausible remedy, ii) articles offering insights into the environmental impact associated with the deployment of extensive Deep Learning models, and iii) articles introducing the techniques underpinning Green AI, aiming at mitigating the cost of Red AI. The outcome emerging from the analysis performed by this work consists of a compromise between sustainability in contrast to the performance of AI tools. Unless the complex training and inference procedures of software models mitigate their environmental impact, it will be necessary to decrease the level of accuracy of production systems, inevitably conflicting with the objective of the major AI vendors. The outcomes of this work would be beneficial to scholars pursuing intricate Deep Learning architectures in scientific research, as well as AI enterprises struggling with the protracted training demands of commercial products within the realms of Computer Vision and Natural Language Processing.

INDEX TERMS Green AI, red AI, survey, environmental impact.

I. INTRODUCTION

The field of Machine Learning (ML) has recently experienced rapid growth and vast recognition, leading to significant advancements during the last few years. For example, Deep learning (DL) has enabled the development of complex

neural networks, leading to breakthroughs in image and speech recognition, natural language processing (NLP), and robotics [1]. Similarly, Reinforcement learning (RL) deployed systems able to play games at advanced levels, control robots, and optimize complex systems [2]. On the other hand, Generative Adversarial Networks (GANs) can generate realistic images, videos, and audio, supporting applications in domains such as design, entertainment, and

The associate editor coordinating the review of this manuscript and approving it for publication was Daniela Cristina Momete .

healthcare [3], while Transfer Learning (TL) enabled the development of more accurate models with fewer data [4]. Finally, AutoML reduced the time and effort required to develop accurate models, also making them more accessible to non-experts [5].

Despite notable progress in ML models' accuracy scores, some Natural Language Processing (NLP) tasks have been solved through models that necessitate significant computational resources, leading to high demand for energy and corresponding financial costs. In particular, considering environmental impact, two distinct approaches to AI research have emerged: Green AI and Red AI.

Schwartz et al. [6] firstly referred to Green AI as a type of Artificial Intelligence (AI) producing both innovative and accurate results without requiring more computational resources or, preferably, by reducing them. Furthermore, the authors stressed that the main objectives of Green AI consist of greater environmental sustainability and the advancement of the AI field responsibly and inclusively. Promoting efficiency instead of accuracy as an efficiency metric is also one of the aims of the authors. In this sense, a balance between performance and efficiency is preferable for sustainability, as it accounts for the computational cost and works towards minimizing the usage of resources.

Although Green AI is a broad concept that encompasses a wide range of AI research, development, and deployment strategies, it is not limited to specific industries or applications. Specifically, the authors discuss that although AI is employed to develop strategies for resolving environmental issues, it is at the same time the (underrated) cause of them. As a result, it is crucial to distinguish between two distinct branches of AI development: Green AI and AI models applied to environmental issues. Although these aspects share the common goal of addressing ecological challenges, their approaches, implementations, and outcomes greatly differ (despite their significance, the topic related to the application of AI models to environmental studies is not in the scope of the present study).

As highlighted in [7], the trend to increase the usage of resources is both prohibitively expensive and damaging to the environment. The employment of Red AI is a problem that cannot be ignored as the increasing energy demand for computation has led to a substantial increase in the size of AI's carbon footprint. Schwartz et al. reports that the following relations emerged:

- Despite the deployment of more powerful hardware resources, the results tend to reach a theoretical limit in terms of accuracy;
- The computational cost increases exponentially;
- In the best scenarios, the relationship between the performance and complexity of a model is logarithmic;
- The main consequence of the usage of Red AI is based on the diminishing returns after increased computational cost over time.

Figure 1 provides a schematic insight into the main motivations that lead to this study. ML is a type of AI,

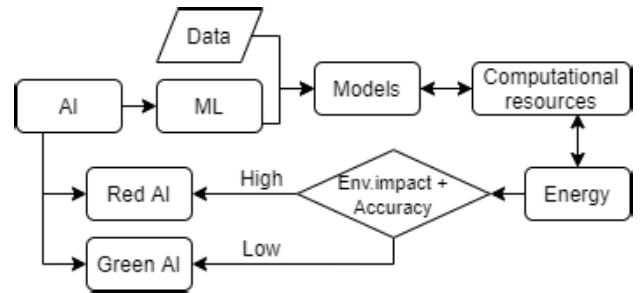


FIGURE 1. Schema of the motivation of the work.

and ML algorithms are trained on input data, leading to the parametrization of models. However, to create such models, computational resources are required, and they are available at the price of energy. The quantity of consumed energy (proportional to the required time and complexity for the training of the model) leads to a certain environmental impact and accuracy. Depending on their measures, the outcome relates to Red or Green AI.

The numerous factors that contribute to the growth of Red AI can be explained in different ways. Despite being a fundamental computing architecture for the last half-century since John Von Neumann first described it [8], it is nowadays not optimized to properly handle the complex processing required by AI formalisms. Models such as Artificial Neuron Networks (ANN) are simulated instead of being executed directly on the machine hardware (i.e. there is no physical correspondence between the machine hardware and the architecture of a neural network). The limitations of the Von Neumann Architecture are becoming apparent as they are leading to slow processing speeds (specifically, the data transfer rate or throughput between the CPU and working memory tends to be limited. In other words, a bottleneck arises when the CPU performs minimal processing on large amounts of data, resulting in high power consumption, and limitations in the size and complexity of supported AI models. Connection machines [9] was the first attempt to create a more performing architecture oriented to AI applications. Another important factor contributing to Red AI's emergence is the intrinsic nature of classic data structures. Non-optimal data structures lead typically to both poor memory usage and reduced computational performance in AI systems. As proper structures provide means to store, organize, and access data, the choice of the right data structure for the task to be accomplished is crucial. When non-optimal data structures are taken into account, AI applications tend to require great amounts of memory (consequently slowing down processing) and lead to inefficient execution of algorithms and processing methods. As a consequence, this inefficiency increases energy consumption. The combinatorial explosion of the model parameters and the number of spatial dimensions originating from the high number of features is known as the curse of dimensionality. Scientific literature provides numerous techniques to mitigate this problem. For example, PCA, t-SNE and UMAP algorithms are widely used in

clustering issues [10] while the autoencoders can reduce the data noise by achieving dimensionality reduction ([11]).

The increasing demand for better-performing models led to the necessity of developing Graphic Processor Units (GPUs) and Tensor Processing Units (TPUs). These units are specialized in performing parallel computing as traditional CPU-based systems have become less efficient in handling the computational workload of AI. Consequently, performing massively parallel computations causes a higher energy consumption and a bigger carbon footprint. The development of DL algorithms accounts for another cause of Red AI, as it is mostly based on sophisticated ANNs architectures, such as CNN [12], RNN [13] and LSTM [14], requiring a massive amount of energy to support their training and inferences.

One of the causes of Red AI is related to profit, as companies often prioritize the development of more accurate ML models to gain a competitive advantage in the market. To obtain such results, speed is a key factor. A shorter computational time leads to faster information gathering and, consequently, quicker decision-making.

Finally, the repetition of experiments based on inferences is a further cause explaining the rise of Red AI. This aspect is often overlooked since research papers only report final results, regardless of all the previous trials. Table 1 summarizes the main differences between Red and Green AI.

TABLE 1. Red and green AI.

Feature	Red AI	Green AI
Training time	Long	Short
Energy consumption	High	Low
Carbon emissions	High	Low
Data requirements	Large	Small
Training methods	Computationally expensive	Efficient
Energy sources	Non-renewable	Renewable

It is worth mentioning that the term Green AI - and more specifically, AI - is somehow vague as, besides stemming from a cultural context belonging to the past century, it gathers several disciplines that are recently conveyed under the ML area. From an epistemological perspective, it can be argued that AI represents a paradigm shift in the way knowledge is understood and approached, while ML is a form of statistical learning mostly driven by data. For example, Domingos [15] argues that ML represents a new kind of scientific inquiry that is data-driven and relies on algorithms rather than theories. In the revised version of an earlier work, Minsky and Papert [16], claim that the traditional approach to AI, which involves building systems that explicitly encode human knowledge and reasoning, has limited success and that ML represents a more promising approach.

While AI and ML are related concepts, they have distinct differences in terms of their scope and application. Pursuing this direction, ML represents a new paradigm for scientific inquiry that is data-driven and relies on algorithms rather than explicit theories.

The impact of the used technologies on sustainability and computational efficiency can be clarified through some real-world examples. Williams et al. [17] study the different impact of the usage of Microsoft's cloud computing Office 365 and traditional Office 2010. Depending on the used MS software, the two solutions perform differently in terms of overall consumed energy. Therefore, preferring one technology to another has an impact on carbon footprint, even for small tasks. As Vishwanath et al. [18] observe that the amount of energy required for cloud services comes from both the operations of data transportation between local and cloud, and by the device that is used by the user. Dropbox service is also being studied by the company itself (https://aem.dropbox.com/cms/content/dam/dropbox/warp/en-us/esg/2021_ESG_Impact_Report.pdf). As it relies on on-demand disposal of data through the usage of cloud, resource efficiency is fundamental for effective energy management. According to the company, the contrast to the increasing size of carbon footprint through the usage of newer and more efficient servers, reduced energy consumption by 25% between 2020 and 2021. Furthermore, the deployment of 100% usage of renewable energy, the introduction of HDD standby policies (which enabled power saving between 25 and 50%), and the eco-sustainable recycling of obsolete devices have provided very good results. Another example of the separation between performance, energy efficiency, and sustainability is given by data centers employed hardware ([19]).

The authors evaluate the performance of green data centers based on ARM and INTEL architectures. INTEL devices are powerful but consume a great amount of energy and produce a high quantity of heat. They are opposed to ARM devices, which are instead less powerful but also less energy-consuming. Hybrid approaches can be implemented, evaluating the trade-off between energy and efficiency.

In the age of climate change and sustainability, it is of paramount importance to harness the power of technology for the improvement of our planet. Green AI represents the intersection of cutting-edge AI techniques and a more mature consciousness. It seeks to create solutions that not only address complex issues but do so with minimal carbon footprints. In a world strongly dependent on energy, the union of AI and sustainability opens up unprecedented opportunities. The motivation of this article serves as a state of the art, guiding scholars and researchers through the dynamic landscape of Green AI. From innovative ML algorithms to sustainable computer hardware, the fusion of artificial intelligence and ecological responsibility is not a mere trend but a defining paradigm of the 21st century. This research aims to inspire, educate, and empower scientists, engineers, policymakers, and citizens alike to take steps toward a more sustainable and environmentally friendly future.

Concerning the limitation of this work, it is important to note that the cost of the contraposition between red AI and green AI could not be studied in depth, as a complete

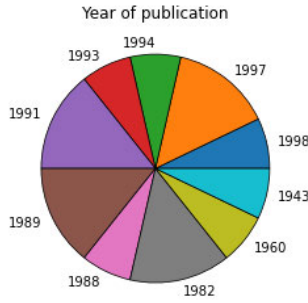


FIGURE 2. Pre-survey: Temporal distribution of foundational articles published before the year 2000.

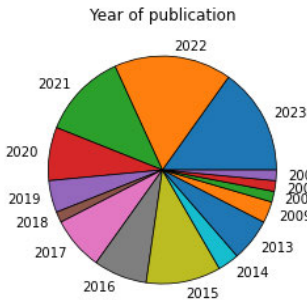


FIGURE 3. Pre-survey: Temporal distribution of foundational articles published after the year 2000.

assessment of a transition from the former to the latter would need a study the terms of a balance between sustainability and performance, which is a rather complex task.

The contribution of this article consists of the discussion of the state of the art of Green AI, and is articulated as follows:

- A review of the factors affecting Red AI, such as computer architectures, data structures, and algorithms performance;
- Notable cases of applications within Red AI, described from the perspective of the time taken by the training, the hardware specifications, and the dataset used;
- A methodology based on a clustering technique to classify the most relevant scientific literature regarding Green AI.

The references used in this work are articulated in the following manner.

As per Figure 2 and 3, references from 1 to 78 (out of 126) concern the sections ranging from I to IV. The references were divided into two figures (articles published before and after the year 2000) for the sake of clarity. Furthermore, 14 articles serving as foundation-cited articles have been published before the year 2000.

However, 33 (out of 78), have been published between 2017 and 2022. These articles consist of a pre-survey: they are considered foundational as they provide the preliminary theoretical background for the following discussion.

The remaining 48 references regard the survey, spread over 7 years (from 2016 to 2023, as per Figure 4); 73% of the articles were published between 2020 and 2022.

The remainder of this work is organized as follows. Section II critically compares similar surveys to this work.

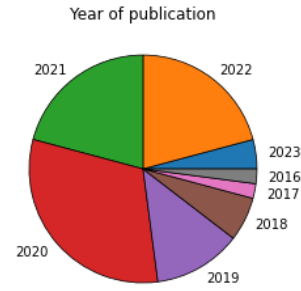


FIGURE 4. Temporal distribution of the articles in the survey.

Section III reviews the main factors contributing to Red AI. Specifically, the discussion delves into different computer architectures, data structures, and algorithms. Section IV introduces the foundations of the most relevant Natural Language Processing and BigGAN models. Section V presents a survey on the Green AI topic according to the problem statement, the energy requested and hardware deployed on training and inference of complex language models, and the approaches aiming to optimize Red AI algorithms. Section VI discusses the trends emerging from the survey and presents some of the lines of future work.

Table 2 provides a short explanation regarding the terms used in sections III and III-C.

Figure 5 provides a flowchart with the proposed methodology that led to the development of the study.

II. RELATED WORK

The subject of green AI is relatively new: as a result, surveys within this discipline are not numerous.

The work presented by Xu et al. [20] is an extended and in-depth review of green DL, articulated in a definition of the topic and the relevant metrics (running time, carbon emission, model size, Floating-Point Operations, fair measure, and intuitive understanding). Interestingly, the term Green AI is equated to Green DL, claiming implicitly an epistemological status of DL. Following this preamble, the authors focus on the architectural aspect, separating the component design from the assembling. The strategies followed by the training algorithms are reviewed in terms of energy-efficient factors (such as initialization, normalization, and progressive training). Similarly, DL is analyzed from the view of the energy requested to perform inferences, that is model pruning, quantization, and distillation. Finally, the authors review the way by which training data can be exploited to make the training phase more parsimonious in terms of resources.

Verdecchia et al. [21] use the Goal-Question-Metric method to clearly state the research question, which sums up in the state-of-the-art of Green AI. The authors thoroughly describe the steps of the research process, then discuss the results of the research, articulated in i) the publications distribution over the period 2015-2020, ii) the venues (conferences are equated to journals and are the majority, opposite to workshops), iii) definitions of Green AI, iv)

TABLE 2. Summary of terms.

Key	Meaning
Connection machines	Parallel supercomputers designed for massively parallel processing using a large number of simple processors interconnected in a network
Distillation	A technique where a larger, more accurate neural network teaches a smaller, less accurate one. This results in a smaller model that performs almost as well as the original. It involves knowledge transfer with three components: type of knowledge, distillation algorithm, and teacher-student architecture
GPU	Graphics Processing Unit. A specialized electronic circuit designed to accelerate the rendering of images and videos on a computer screen.
High-Performance Computing	It refers to the use of powerful, specialized computer systems to process and solve complex problems, typically involving large-scale data and computations, at speeds significantly faster than traditional computers.
MobileNet	The MobileNet architecture employs depthwise separable convolution to create efficient and lightweight convolutional neural networks for mobile and embedded devices. This technique decomposes weight matrices into low-rank matrices, reducing parameters and computational complexity.
Model binarisation	The process of converting the weights and parameters of a neural network into binary values (usually 0s and 1s) to reduce memory and computational requirements, often used for deploying deep learning models on resource-constrained devices.
Neuromorphic Computing Architecture	Neuromorphic computing architecture is a design inspired by the human brain, using artificial neural networks and analog circuits to mimic the brain's information processing. It aims to improve efficiency and handle tasks like pattern recognition and sensory processing.
NTM	Neural Turing Machines are neural network models equipped with external memory and read-write operations, enabling them to perform tasks that involve memory, making them more versatile for complex sequential tasks.
Quantum Computer	A type of computer that uses principles of quantum mechanics to perform computations. It has the potential to solve certain complex problems much faster than classical computers.
RBFN	Radial Basis Function Network. It's a type of artificial neural network that uses special functions called radial basis functions to process data. RBFNs are often used for tasks like pattern recognition and function approximation.
Residual Connection	A shortcut for information to flow through the network. They help in training very deep neural networks by allowing some of the input to directly reach deeper layers. This makes it easier for the network to learn and can lead to better performance.
RNN	A Recurrent Neural Network is a type of computer program that's good at working with sequences of data. It's like a smart computer that can remember and use what it saw or learned in the past to help it understand and predict what comes next in a sequence, like in language, time series, or music.
SOM	A Self-Organizing Map is a type of artificial neural network that helps organize and visualize complex data by mapping it onto a grid. It's often used for tasks like clustering and dimensionality reduction.
SNN architecture	A Spiking Neural Network is a type of artificial neural network that models information processing in the brain using "spikes" or brief pulses of activity. It's used to simulate more biologically realistic neural behavior and is often employed in neuromorphic computing.
Structural matrices	Structural matrices are matrices that describe the connections or relationships between elements in a system, often used in various fields like graph theory, finite element analysis, and network science to represent the underlying structure of a complex system.
Tensor	A mathematical object that represents multi-dimensional data, such as scalars (0D), vectors (1D), matrices (2D), and higher-dimensional arrays. Tensors are used in various fields, including physics and machine learning, to handle and process complex data structures.
TPU	A Tensor Processing Unit is a specialized hardware accelerator developed by Google for machine learning tasks. It's designed to efficiently perform matrix and tensor operations, making it well-suited for deep learning and neural network workloads.
TrueNorth Architecture	A neuromorphic computing design developed by IBM. It emulates the brain's structure and function, enabling energy-efficient and parallel processing for cognitive computing tasks.
Von Neumann Bottleneck	A limitation in traditional computer architecture where the speed of data transfer between the CPU and memory is significantly slower than the CPU's processing speed. This can slow down overall system performance.
Weight sharing	Weight sharing in deep learning involves using the same set of weights and biases for multiple units or connections within a neural network layer. It helps reduce the number of parameters and promotes feature reuse, making the network more efficient and effective.

study types, v) Green AI topics, subdivided into sub-topics (such as hyper-parameter tuning, model benchmarking,

deployment, model comparison, etc.), vi) Green AI Topics presented by Study Type, vii) domains (such as edge

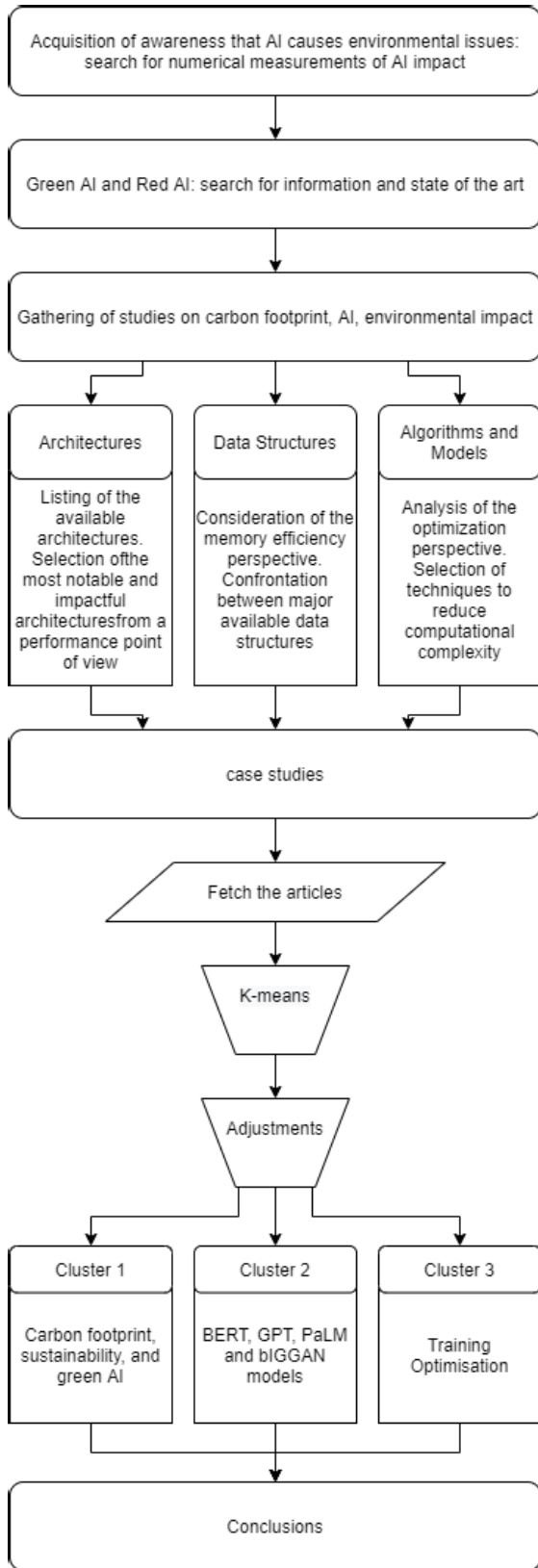


FIGURE 5. Study organization.

computing, computer vision, cloud, and mobile computing), viii) AI pipeline phases (training, inference, or the whole

pipeline), ix) artifacts (such as model, data, pipeline), x) algorithms types, xi) data types considered (image, text, numbers, video or audio data), xii) the size of the datasets, xiii) research strategies (distinguishing between experiments performed in a laboratory, on the field, or by simulation), xiv) energy-saving evidence, xv) industry involvement (most of the studies originated from universities rather than the industry), and finally xvi) intended readers (mostly from the academy).

While Verdecchia et al.'s article is an accurate bibliometric study meant to provide a state-of-the-art, the current work is closer to XU et al., as it is meant to provide the foundations of the problem, the deployed DL architectures, data and algorithms used, also providing a clusterization of the scientific literature about Green AI.

As AI is rapidly increasing its impact on both technology and society, IEEE has been working on creating comprehensive standards encompassing a spectrum of AI applications. Although there is a growing emphasis on Green AI against Red AI, the up-to-date standards mainly focus on the major complex ethical, environmental, and technical challenges that AI presents. Salehi and Schmeink [22] discuss the importance of data in relationship with AI and sustainability, concluding the need for data benchmarks rather than model benchmarks when working on data-centric AI. The authors provide 36 benchmarks, considering different tasks, goals, domain of application and state of their retrieved data. Siegmund et al. [23] focus on techniques for reducing energy consumption using AI in software systems. As a significant amount of energy wasted for computation can be saved by optimizing the choice of the parameters, the authors suggest AI and ML methods for finding more efficient configurations. Unfortunately, the fact that black-box models do not provide meaningful insights represents an important drawback. Again, Gutierrez et al. [24] underline the need for developing ML models that simultaneously meet the expected operational requirements and guarantee a balance between obtained results and energy consumption. In conclusion, IEEE authors strongly suggest increasing the effort in software development, as architectural choices strongly affect energy consumption.

III. MAIN FACTORS CONTRIBUTING TO ENERGY CONSUMPTION

Assessing the amount of energy requested by the execution of a computational process is not an easy task. The evaluation of the cost deriving from the training of DL models requires broader considerations, including the computer architecture and the hardware, the training algorithm, and the type of employed data structures. Following this direction, this section reviews i) the most notable classic architecture in chronological order, ii) data structures and models typically deployed in DL. Finally, the section reviews the quantitative approaches in literature providing a numerical estimate to train complex DL models.

A. COMPUTER ARCHITECTURES

1) 1940 - 1950

The Von Neumann architecture is characterized by a unified memory structure for instructions and data, which is shared through a single data bus and address bus between the processor and memory. Access to instructions and data is subject to sequential order, giving rise to the so-called von Neumann Bottleneck [25]. This restricts the memory access bandwidth and results in idle processor cycles during memory access operations. Despite the advent of various solutions such as cache memory and branch predictor algorithms in modern computer architectures, these novel techniques have not completely eradicated the limitations imposed by Von Neumann Bottleneck.

The Harvard architecture [26] uses separate memory spaces for instructions and data. The instruction and data buses are physically separate, allowing instructions and data to be fetched and processed simultaneously. This can improve performance in certain applications, but may also require more complex programming techniques. The modified Harvard architecture [27] combines elements of both von Neumann and Harvard architectures. In this proposal, instructions and data are stored in separate memory spaces, but the processor can also access instructions stored in the same memory space as data.

2) 1950 - 1980

High-Performance Computing (HPC, [28]) Systems are typically used in supercomputers and data centers and are designed to handle massive workloads and perform complex calculations. Due to their high processing power, they typically consume substantial amounts of energy. However, advancements in energy-efficient designs, such as power management techniques and specialized processors, have mitigated this to some extent. GPUs are highly parallel processors primarily used for graphics rendering and accelerating computationally intensive tasks. While they excel at parallel processing, their power consumption can be relatively high due to a large number of cores and memory bandwidth requirements. Modern GPUs are becoming more power-efficient, but high-end gaming GPUs can still consume considerable energy. Custom Application-Specific Integrated Circuits are specialized chips designed for specific tasks, such as cryptocurrency mining or deep learning inference. While ASICs can offer significant performance gains, they often prioritize performance over energy efficiency. ASICs consume substantial amounts of energy due to their intensive computational requirements. Traditional INTEL-based CPUs have undergone significant energy efficiency improvements over the years. However, high-performance desktop CPUs can still consume substantial power, especially during demanding tasks like gaming or video rendering. Server-grade CPUs designed for data centers (or even heterogeneous computing architectures based on both INTEL and ARM architectures) can also consume significant energy due to their processing power and scalability [29].

Quantum computers (QCs) represent a different paradigm in computing and must operate at extremely low temperatures, often near absolute zero. Scientific literature concerning this type of architecture is too wide to be analyzed in this work, although a recent analysis of the current trends can be found in [30]. Achieving and maintaining these temperatures requires specialized cooling systems, such as cryogenic refrigerators. They consume significant energy to create and sustain the required infrastructure [31]. QCs rely on precise control of individual quantum bits (qubits). This necessitates complex control systems that manage the interactions and manipulations of qubits. These control systems can consume energy, especially when addressing large numbers of qubits at the same time. As QCs are susceptible to errors caused by environmental disturbances and imperfect qubit operations, implementing quantum error correction techniques requires additional qubits and computational overhead. As a result, this leads to increased energy consumption. Furthermore, QCs involve applying operations known as quantum gates to qubits. The energy consumption associated with performing quantum gate operations can depend on the implementation and the specific architecture of the quantum computer. Finally, extracting information from qubits through measurement processes and measurement operations often involve amplification and signal processing, which require additional energy.

Although outdated, the Spiking Neural Networks (SNNs) architecture [32], proposed in 1997, extends the concept of ANNs. By identifying the first ANN generation with Rosenblatt's perceptron, Hopfield nets, and Boltzmann machines, the second generation can be denoted by models taking into account sophisticated activation functions (such as the sigmoid functions). In this sense, it can be noted that a third ANN generation consists of computation units working as spiking neurons. The advantage of this proposal emerged in providing models of boolean functions deploying an inferior number of gates with respect to the previous ANN generations.

3) 1980 - 2000

CNNs [33] are partially based on ANNs as their basic component is a cell connected to its neighbors. However, a cell that has no connections can be indirectly influenced because of the continuous-time dynamics of the network. CNNs semantics and evolution can be mathematically described in terms of ordinary differential equations and be physically realized in the shape of operational amplifier-based circuits. Typical applications of CNNs revolve around image processing and signal processing in real-time.

Self-Organizing Maps (SOMs [34]) is based on the idea that a 1- or 2-dimensional array represents the correct maps of structured distributions of signals, despite not having this exact structure initially. The processing units by which the mappings are realized, share some similarities with Rosenblatt's perceptron [35] and are loosely based on maps of sensory experiences located in the brain. The

core of the self-organizing process of SOMs rests on i) an array of processing units composing simple discriminant functions, ii) a tool comparing the discriminant functions and choosing the greatest value returned by the function, iii) a phase where the selected unit and its nearest neighbors are activated at the same time and finally iv) a process that adaptively increases the discriminant function values of the parameters of the activated units. The work of Hopfield [36] aims at describing emerging computing properties in a neural network where every single neuron has very limited capabilities. Nevertheless, it is possible to observe a *gestalt* behavior able to manifest generalization properties and the capability to order memories concerning time.

The interest around RBFNs [37] stems from the ability to approximate arbitrary functionals of a finite number of real variables. In this sense, RBF networks consider one hidden layer to perform universal approximation.

Recurrent Neural Networks (RNNs, [38]) can perform durable data transformation, have a strong learning capability, and are Turing complete, meaning they can compute every Turing-computable function. Alone, RNNs show a deficiency: their limited memory causes both computed and input data storage difficulties. Therefore, the availability of a higher quantity of memory for RNNs is crucial as it enables a stronger understanding of the relationships between input elements, as stressed by Suresh et al. [39]. Furthermore, merging TM and RNNs allows for a new architecture that is memory augmented as RNNs own the equivalent infinite memory tape as TM. By construction, RNNs process sequential data using feedback connections, to persist information that is processed over time at a cost of a significant increase in computational complexity compared to simpler models, such as feedforward neural networks. RNN are usually deployed to elaborate long input sequences triggering a large number of recurrent operations, leading to increased energy consumption. The longer the sequence, the more computations are needed, resulting in higher energy requirements. Training RNNs involves back-propagation through time (BPTT), which requires computing gradients for each time step in the sequence. This process can be computationally demanding and resource-intensive, leading to increased energy consumption during training. Larger RNN models with more parameters require more computational resources and, consequently, more energy to train and run. Finally, the NTM architecture consists of a controller and an external memory. Different from TM, NTM includes a more efficient learning mechanism.

4) 2000 - 2020

LSMs [40] are built upon the principles of high-dimensional dynamical systems paired with statistical learning theory. This architecture can be exploited to implement a universal analog fading memory thanks to the inherent transient dynamics of the high-dimensional dynamical system originated by a heterogeneous neural circuit sufficiently large.

Graves et al. [41] introduced a fundamental evolution of Turing Machines, Neural Turing Machines (NTMs), which combine TM and NN. The improvements of NTMs followed different directions, articulated in Reinforcement Learning Neural Turing Machine [42], Evolving Neural Turing Machine [43], Lie Access Neural Turing Machine [44], Dynamic Neural Turing Machine [45], and Neural Random Access Turing Machine [46]. NTM is a novel architecture that combines the characteristics of TM and RNNs.

The TrueNorth Architecture [47] is a novel non-Von Neumann Architecture proposed by IBM researchers. Drawing inspiration from the human brain, this architecture employs a memory controller akin to a neuron, while eschewing a central processing unit. Notably, it is designed to operate on synaptic-like datasets stored in memory, without any data rearrangement, by leveraging the crystallization dynamics of phase-change memories. Furthermore, a plethora of alternative architectures have emerged in contemporary times. For example, the Neuro-morphic Computing Architecture [48] aims at developing a computing framework, based on IBM's Blue Gene supercomputer (<https://www.ibm.com/ibm/history/ibm100/us/en/icons/bluegene/>) able to simulate cortical circuits by using Artificial Neural Networks.

Table 3 summarizes the different computer architecture evaluations, while fig. 6 shows a timeline.

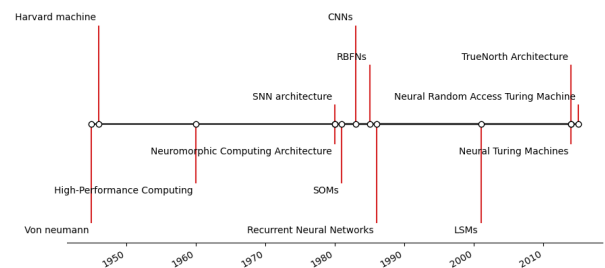


FIGURE 6. Computer architectures timeline.

B. DATA STRUCTURES

Some data structures are more memory-efficient than others. By selecting data structures that use memory efficiently, it is expected to reduce the energy required for memory operations such as reading, writing, and moving data. For example, using a linked list instead of an array can save memory when working with dynamically changing data. Different data structures offer varying performance characteristics for different operations. Choosing data structures with efficient algorithms can lead to faster execution times, which can result in reduced energy consumption. For example, using a hash table for efficient lookup or a balanced binary search tree for efficient searching can reduce the computational workload and energy usage. Furthermore, the choice of data structure can affect factors such as cache utilization and data locality. Data structures that exhibit good spatial and temporal locality can reduce the number of memory

TABLE 3. Most notable computer architectures.

Architecture	Year	Evaluation	Reason
Harvard architecture [26]	1946	Poor	It can improve performance in certain applications, but may also require more complex programming techniques
High-Performance Computing [28], [29]	1960	Poor	Due to their high processing power, HPC typically consumes substantial amounts of energy
Quantum computers [30] [31]	1979	Medium	Radical paradigmatic change in the concept of computing, although QC presents cooling issues
Recurrent Neural Networks [38], [39]	1986	Poor	RNNs are energy expensive and require a higher quantity of memory
TrueNorth Architecture [47]	2014	N/A	Architecture closely based on the human brain's organization and ANNs
Evolutions of Turing machines (such as Neural Turing Machines [41], Neural Random Access Turing Machine [46])	2014	High	Reduced Network Complexity, efficiency of planning and control tasks, computational efficiency and speed up processing compared to solely sequential models

accesses and improve cache hit rates, thus saving energy by minimizing data transfer between levels of the memory hierarchy. In a similar way, dynamic arrays or resizable hash tables may require resizing or reallocation as the data size changes. These operations can be computationally expensive and consume energy. Choosing data structures that minimise the frequency of resizing or optimize the resizing process. Finally, using specialized data structures suitable to the task at hand can result in energy savings. For example, spatial data structures like quadtrees or octrees can efficiently handle spatial queries. With regard to ML, the most well-known data structures are vectors. Vectors were used initially in physics to represent various quantities such as velocity, acceleration, force, and momentum, although they were formally defined in linear algebra by Sir William Rowan Hamilton, with regard to vector fields. Furthermore, vectors are used in biology to represent the direction and magnitude of various biological processes such as cell movement and protein folding. Nowadays, vectors are central in ML as they can be used for the following tasks:

- Data representation, where each vector represents a sample or observation;
- Feature extraction in text classification, where words can be represented as vectors using techniques such as word embeddings.
- Model training, as vectors represent the input data to the model, as well as the model's parameters and output.
- Similarity and distance measures in recommendation systems, as vectors representing user preferences are compared to vectors representing item features to identify those items that are similar to the user's preferences.
- Optimisation to represent the variables being optimized, such as the weights of a neural network.

In ML, storing information in a vector is beneficial for multiple reasons. Firstly, storing numerical data in a vector allows easier processing and comparison between entities, resulting in a convenient way to store and manipulate data. In fact, while memorising datasets' content in their original form can be memory-intensive, representing them as vectors reduces the size of data and allows for both easier and better

storage and manipulation. In ML modeling, Support Vector Machines (SVM) obtain better classification via the usage of computation of data stored in vectors, as it finds the optimal hyperplane through the analysis of vectors as n-dimensional space. Furthermore, from a computational point of view, vectors paired with high-performance scientific scripting languages - such as MATLAB or Python Numpy - enable the more efficient application of mathematical operations onto large amounts of data. Moreover, the usage of vectors also comprises efficient computation through the usage of adequate hardware: GPUs are designed to perform vectorized operations, and hence they can perform complex computations on large datasets much faster than a CPU. A further strength consists of the fact that mathematical computations can be performed on entire vectors simultaneously through vectorized operations. Vectors present some weaknesses as well, the main of which is that the use of high-dimensional vectors can lead to sparser data, thus leading to uniform distance between data points. Consequently, ML algorithms that measure similarity or distance between the given data points tend to perform poorly. The described issue is known as the curse of dimensionality (see, for example, [49]).

The usage of matrices spans from computing linear transformations and solving linear equations to representing similarity or distance between samples or features.

The notion of a matrix was first formulated in the middle 1800s, but it is nowadays enormously exploited for various types of mathematical applications. In ML, the size of the input matrix impacts significantly the performance of the considered models. Sparse matrices occur frequently in large-scale ML applications, resulting in reduced memory usage compared to dense matrices. This is especially beneficial when dealing with high-dimensional datasets or large feature spaces. By avoiding the storage of numerous zero values, sparse matrices conserve memory and allow models to fit into memory more easily, enabling larger datasets to be processed. Also, sparse matrices can accelerate computations in ML models by skipping the multiplication or addition operations involving zero values. This can lead to faster execution times and improved overall model efficiency. Several ML algorithms are specifically

designed to take advantage of sparse matrices. For instance, sparse implementations of algorithms like logistic regression, support vector machines (SVMs), or collaborative filtering can be significantly faster and more memory-efficient than their dense counterparts. These algorithms leverage the inherent sparsity of the data to optimize computations and reduce memory requirements.

Additionally, matrices provide a compact, flexible, and efficient representation of high-dimensional data. As a result, performing linear algebra operations on the entire dataset at once is computationally efficient and avoids redundant computations. Matrices are easily extendable to accommodate new data or features, making them a valuable tool for many ML applications. Another essential advantage is that, as they offer a wide spectrum of available algebraic operations, matrices are a powerful mathematical tool that allows efficient processing, manipulation, and analysis of high-dimensional data. Finally, matrix operations can be easily executed in parallel, leading to efficient implementation on parallel architectures such as GPUs. It has to be noted that large and high-dimensional non-sparse matrices can entail potential issues for memory storage, computation, and optimization. Consequently, matrix operations on large-scale datasets can be computationally expensive, limiting both the scalability and performance of ML algorithms. Matrix operations can be numerically unstable, particularly when dealing with ill-conditioned matrices, such as those with nearly zero eigenvalues.

The notions of vector and matrix are generalized by the tensor's concept. Although they originated in the first years of the 1900s, tensors have lately become a fundamental data structure for ML.

A tensor is a multi-dimensional collection of data, can contain different data types, and can be exploited by ML algorithms. Kolda and Bader [50] state that an N th-order tensor can be defined as an object that belongs to the tensor product of N vector spaces, where each vector space has its own coordinate system. Tensor decomposition (Rabanser et al. [51] and Ji et al. [52]) is a fundamental tensor operation for ML applications that aims to reduce the complexity in terms of the amount of time and computational resources required to perform a specific task. In general, the operation consists of decomposing a tensor into the minimum sum of tensors of rank 1 (the so-called decomposable tensors). The result is the decomposition of a high-dimensional tensor into smaller tensors which are easier to manage and work with. The principal advantages consisting of reduced execution time, more parsimonious memory utilization, and better scalability. Based on the fact that a tensor decomposition is independent of the coordinate system, Bernardi et al. [53] provides the means to the extraction of the given data's geometric or invariant properties. Effective and efficient Artificial Neural Networks ANNs are built of tensors. In fact, the usage of tensor as ML data structure brings multiple advantages. Firstly, it provides a structured way to represent data in a consistent

and organized manner, therefore it is a convenient data format. In fact, it is possible to use a tensor as an efficient way to organize multi-dimensional arrays. Secondly, it allows for very efficient operations, consequently speeding model training and inference. A tensor is compatible with various mathematical operations and transformations. Moreover, it reduces complexity in an effective way, as organising information in a tensor allows for easier processing and manipulation and enables efficient implementation of mathematical applications and transformations. Furthermore, it is a way to represent data, independently of basis and coordinates, isolating intrinsic geometrical and physical properties from coordinates' dependant properties. Representing data in a coordinate-free manner allows us to concentrate on the intrinsic properties of data rather than specific characteristics of data. Last, it allows for solving high-dimensional problems, as it simplifies the representation, manipulation, transformation, and processing of high-dimensional data, therefore make it possible to suit the specific needs of an application. The first drawback is that tensor decomposition requires a high number of parameters and samples to improve accuracy. Consequently, if a non-fully-optimized algorithm is used, the increasing number of required parameters makes convergence very slow or even impossible to reach. The lack of convergence can occur also due to the wrong initialization of tensor operations. Furthermore, due to a combinatorial explosion, the choice of promising features tends to be rather complex. As an example, figure 7 shows a vector of dimension 4, a 4×4 matrix and a $4 \times 4 \times 4$ tensor.

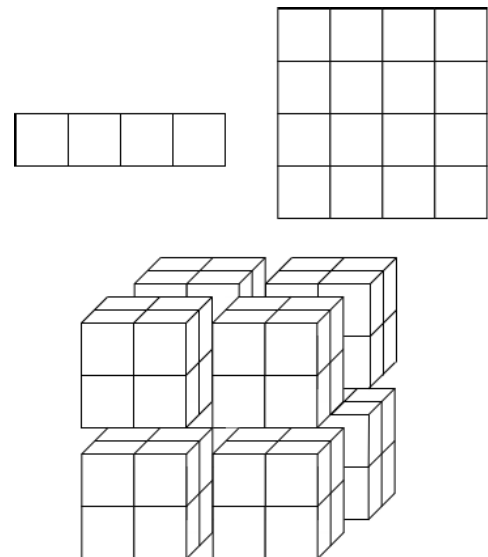


FIGURE 7. A vector, a matrix and a tensor.

The use of tensor processing units (TPUs) can potentially contribute to energy efficiency in certain scenarios. TPUs (see, for example, [54]) are specialized hardware accelerators designed by Google for performing tensor operations efficiently. They excel at handling large-scale tensor computations, which are prevalent in DL tasks.

The energy efficiency of TPUs stems from their ability to parallelize and optimize tensor operations, resulting in faster and more efficient computations. By leveraging TPUs, computational tasks involving tensor operations can be executed with higher throughput and lower latency compared to traditional CPUs or GPUs. This increased efficiency translates into potentially lower energy consumption for the same computational workload.

C. ALGORITHMS

Although the models considered in DL have been proven to be scalable [55], they still require sophisticated hardware architectures to run, resulting in a drawback when they are deployed to mobile devices. As a result, scientific literature has provided different techniques aiming at producing smaller although equivalent models to the initial one. In this direction, this section provides an overview of the most notable approaches to reduce the computational complexity of a model training phase.

Distillation [56] consists of a technique in which a larger and more accurate neural network is used to teach a smaller and less accurate neural network. The latter learns to mimic the behavior of the larger network, resulting in a smaller model that performs nearly as well as the original, larger model. According to the authors, a distillation system can be regarded as a knowledge transfer problem. It consists of the following components: a type of knowledge, which can be one of response-based knowledge, feature-based knowledge, and relation-based knowledge; a distillation algorithm; and a teacher–student architecture. The distillation process is subdivided into three schemas, namely, offline, online, and self-distillation. In the offline schema, the teacher model is initially trained, and then the knowledge is extracted in the shape of logits, which serve the objective of training the student model. However, due to some shortcomings, an online scheme is preferable. In this case, both the teacher and the student model are updated at the same time. Finally, in the self-distillation scheme, the student and the teacher use the same networks.

Residual connections [57], also known as skip connections, is a technique used in DL architectures to help overcome the vanishing gradient problem. This problem arises when training deep neural networks, as the gradients of the loss function can become very small as they are backpropagated through the many layers of the network, leading to slow or even stagnant training. Residual connections work by allowing the input of a layer to be added directly to the output of a later layer, effectively “skipping over” some of the layers in between. This can help to preserve the original signal and ensure that gradients can flow more easily through the network, which can lead to better performance. Residual connections can help reduce a DL model by allowing for deeper and more complex architectures without sacrificing performance or increasing the risk of overfitting. In traditional deep neural networks, each layer is required to learn a unique representation of the input data. This can

lead to redundancy and inefficiency in the model, as multiple layers may be learning similar features. Additionally, as the number of layers in the network increases, the problem of vanishing gradients can become more severe, making it difficult to train the model effectively. By using residual connections, a DL model can be made more efficient by allowing information to bypass certain layers and propagate directly to the later layers in the network. This can help to reduce redundancy in the model and make it easier to train, as gradients can flow more easily through the network.

Depthwise separable convolution is a type of convolutional operation used in deep learning models to reduce the number of parameters and computational complexity of a neural network. In a traditional convolution operation, each filter (or kernel) is convolved across all input channels to produce one output channel. This can result in a large number of parameters and slow computation when dealing with high-dimensional inputs, such as images. On the other hand, depthwise separable convolution splits the traditional convolution operation into two separate layers:

- Depthwise convolution: In this step, a single filter is convolved separately with each input channel. This produces a set of output feature maps with the same number of channels as the input.
- Point-wise convolution: In this step, a 1×1 convolution is applied to the output of the previous layer. This combines the features from the depthwise convolution layer and produces the final output.

In [58], Howard et al. present the MobileNet architecture, which uses depthwise separable convolution to create efficient and lightweight convolutional neural networks for mobile and embedded devices.

Weight sharing [59] refers to the practice of using the same set of parameters (that is, weights) for multiple parts of a neural network. This is often done to reduce the number of parameters required to train the network and to encourage the network to learn shared representations of the input. For example, in convolutional neural networks (CNNs), weight sharing is used to apply the same filter (i.e., set of weights) to different parts of the input image. This allows the network to learn features that are relevant across different regions of the image and also reduces the number of parameters required to learn these features. Another example of weight sharing is in recurrent neural networks (RNNs), where the same set of weights is used for each time step of the sequence. This allows the network to learn a representation of the input sequence that can be used to make predictions at each time step.

Cheng et al. proposed an article that discusses acceleration techniques for deep learning models. These techniques can be categorized into three main types: hardware-based acceleration, software-based acceleration, and algorithmic acceleration. Hardware-based acceleration involves the use of specialized hardware such as Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), or Application-Specific Integrated Circuits (ASICs) to accelerate the computation of deep learning models. GPU

are commonly used for training deep neural networks, whereas FPGAs and ASICs are more specialized for specific applications such as computer vision or speech recognition. In contrast to the former, software-based acceleration involves optimizing the software implementation of the DL models to improve their performance. Examples of software-based acceleration techniques include parallelization, which involves splitting the computation across multiple processing units, and vectorization, which involves performing computations on multiple data elements simultaneously. Finally, Algorithmic acceleration involves modifying the architecture or algorithm of the DL model to reduce its computational complexity or increase its efficiency. Examples of algorithmic acceleration techniques include pruning, quantization, and low-rank approximation, which were discussed in the context of model compression.

These methods focus on removing inessential parameters from deep neural networks without any significant effect on performance. This category is further divided into model quantization [60] and model binarisation [61]. The authors propose a technique called BinaryConnect, which uses a binary approximation of the weights during both forward and backward propagation. This binary approximation allows for significant reductions in memory usage and computational complexity, while still maintaining high levels of accuracy. The authors demonstrate the effectiveness of their method on a variety of tasks, including image classification and speech recognition, and show that it can achieve performance comparable to full-precision networks, and parameter sharing. Finally, structural matrices are discussed by Sindhvani et al. [62] by introducing a novel method to reduce the memory and computational requirements of DL models by using structured weight matrices. The authors introduce a family of structured transforms, called Structured Matrices (SMs), which can be used to impose various types of structure on the weight matrices of deep neural networks. The authors demonstrate the effectiveness of their approach on a variety of tasks, including speech recognition and image classification, and show that it can achieve significant reductions in memory usage and computational complexity without sacrificing accuracy. Parameter pruning involves identifying and removing the least important parameters (i.e. weights) from a neural network based on some criterion such as magnitude, connectivity, and sensitivity. The goal is to reduce the number of parameters in the network without significantly affecting its performance. This can be done by setting the values of the pruned weights to zero or by completely removing the corresponding connections from the network. This technique is commonly used in DL to improve the speed and memory efficiency of the network and to reduce overfitting.

Krizhevsky [63] focus on iterations of mini-batch stochastic gradient descent (SGD), the mostly used algorithm employed in the training of neural networks. The key idea is to involve successive modifications to the model's parameters through an approximation of the gradient pertaining to the

training objective. This gradient estimation is performed at each iterative step utilizing distinct subsets, commonly referred to as mini-batches, extracted from the larger training dataset. The authors refer to Data parallelism as a widely used approach to accelerate neural network training. It involves distributing training examples among multiple processors to compute gradient updates or higher-order derivatives, and then aggregating these updates locally. This method is effective for any neural network architecture as long as the training objective can be expressed as a sum of training examples. On the other hand, the extent of model parallelism, which involves distributing parameters and computations across processors for the same training examples, varies based on the model's size and structure. While data parallelism is easier to implement, larger-scale systems should take advantage of various types of parallelism for optimal performance. The most interesting contribution of this work consists of an evaluation of the costs and benefits of adopting this technique in the synchronous training setting, achieving the following conclusions. Firstly, the correlation between batch size and the required number of training steps to achieve a desired out-of-sample error follows a consistent pattern across six distinct neural network categories, employing three training algorithms and encompassing seven distinct datasets. Secondly, the optimal batch size, which is most effective, varies notably across different workloads and is influenced by attributes of the model, training algorithm, and dataset. Thirdly, the ideal settings for training meta-parameters do not consistently adhere to straightforward associations with the batch size. Specifically, widely used learning rate guidelines, like proportionally adjusting the learning rate based on batch size, do not universally apply to all challenges or batch size configurations. Finally, it is possible to mitigate some of the contradictions present in the literature regarding the impact of larger batch sizes on model performance. Table 4 summarizes the algorithms discussed in this subsection.

IV. CASE STUDIES

A. INTRODUCTION

DL models have a measurable effect on the environment, mainly because training and inferences are performed by the model. The entity that is usually measured is the carbon footprint. According to Wiedmann and Minx [64], this term has been characterized in different forms, although their work conveys the following definition: "The carbon footprint is a measure of the exclusive total amount of carbon dioxide emissions that is directly and indirectly caused by an activity or is accumulated over the life stages of a product". The authors identify the methodologies apt to calculate the carbon footprint in a bottom-up Process Analysis (PA) and a top-down Environmental Input-Output (EIO) approach. In general, there is a lack of consensus concerning the metrics that need to be adopted in measuring the footprint. Furthermore, measuring the footprint derived from AI applications may require different methods from those deployed on generic computationally based applications. The authors

TABLE 4. Most notable training algorithms.

Algorithm	Description
Distillation	A technique in which a larger and more accurate neural network is used to teach a smaller and less accurate neural network
Residual connections	It helps to overcome the vanishing gradient problem.
Depthwise separable convolution	It supports the reduction of the number of parameters and computational complexity of a neural network
Weight sharing	It reduces the number of parameters required to train the network and to encourage the network to learn shared representations of the input
Acceleration techniques	A mix of hardware-based acceleration, software-based acceleration, and algorithmic acceleration
Parameter pruning	Removing inessential parameters from deep neural networks without any significant effect on performance
Mini-batch stochastic gradient descent	To implement successive modifications to the model's parameters through an approximation of the gradient of the training objective

stress the importance of evaluating the cost of the operational energy. This consists of the production, transport, and end-of-life stages, added to produced greenhouse gas. The problem of carbon footprint originating by computationally expensive applications is discussed in the following section by taking into account a few case studies related to Natural Language Processing (NLP).

B. NLP MODELS

According to Jones [65], the birth of Natural Language Processing (NLP) goes back in time, at least to 1950, although this study area was known as Mechanical Translation (MT). Initially, MT revealed very promisingly, to the point of indulging scholars in predicting extraordinary results within the next decade. However, the analysis and understanding of natural language is a task that was revealed to be more complex concerning what was initially imagined, as more interdisciplinary competencies and concepts - such as Chomsky's syntactic structures - had to be considered. More sophisticated AI-based approaches were introduced starting from 1970 (such as Schank's Conceptual Dependency Nets leading to an MT revival in the shape of NLP, which pursued new challenges, such as retrieving information from vast amounts of data. More recently [66], ML determined a paradigmatic leap in favor of neural models, and more specifically, sophisticated DL architectures. According to the authors, DL still presents a few challenges in NLP domain that need more investigation, such as the lack of labeled data, a problem mitigated by adopting semi-supervised learning approaches.

Complex language models such as BERT, XLNet [67] and T5 (Text-to-Text Transfer Transformer) [68] are densely overparameterised. As a result, their training is computationally intensive and requires a long time. Furthermore, the training of language models requires sophisticated hardware, such as premium GPU-enabled NVIDIA DGX workstations (maximum power draw: 1,500W, <https://docs.nvidia.com/dgx/pdf/DGX-Station-Site-Preparation-Guide.pdf>) or specialized accelerators such as Google's TPU Pods (Tensor Processing Units), i.e. Google's custom-developed application-specific integrated circuits (ASICs) deployed to accelerate ML workloads (<https://cloud.google.com/tpu/docs/tpus>). Bidirectional Encoder Representations from Transformers (BERT) is a language representation model introduced by Google

(a discussion of the principles ruling BERT training can be found in [69]). Its architecture consists of a pile of Transformer encoders [70] layers. In turn, each one is composed of different self-attention "heads", whose tasks can be summarised as per the following. Firstly, an input token is detected. Secondly, a head is triggered to compute a pair (key, value,) and a set of query vectors implementing a weighted representation. Each head produces an output, which is conveyed with others into the same layer and finally runs through another layer. Pre-training and fine-tuning are fundamental operations in BERT serving different purposes (to predict the next token and the next sentence respectively). Specifically, MLM (Masked Language Modeling) is a pre-training task exploited to train BERT on vast amounts of unlabelled text data. The goal is to mask (i.e., randomly replace) some of the tokens in the input sentence and then train the model to predict the original token according to the context provided by the other tokens in the sentence. During training, BERT replaces 15% of the input tokens with a special token, and then it is trained to predict the original token from the remaining part of the context. By predicting the masked words, BERT learns contextualised word representations capturing the meaning and relationships of words within a sentence. The second task is represented by Next Sentence Prediction (NSP) predicting whether two sentences are consecutive or not. This step supports BERT in understanding the relationships between different sentences.

Chen et al. [71] discusses the concept of the Lottery Ticket Hypothesis (LTH), introduced by Frankle and Carbin [72]. It is a concept in the field of neural network training and optimization and suggests that within a densely initialized neural network, there exist "winning tickets" or subnetworks with a sparse subset of connections that, when trained in isolation, can match the performance of the fully connected network. In other words, LTH's architecture implies that even though neural networks are often overparameterised with many more connections than necessary, it is possible to identify a much smaller subnetwork that can achieve comparable performance if properly trained. These "winning tickets" are essentially initializations of the network's parameters that, when combined with the right training procedure, lead to the rapid convergence of the network to a highly accurate solution. The LTH has significant implications for the optimization of neural networks. It suggests that during training, not all parameters or connections are equally

important, and a systematic pruning approach can be used to identify and retain only those connections that contribute significantly to the network's performance. This can lead to more efficient and faster training, as well as reduced memory and computation requirements. This approach is generally used in Computer vision and NLP. The authors propose EarlyBERT, a novel method that is capable of identifying structured winning tickets during the initial phases of BERT training. These identified winning tickets are then effectively utilized for streamlined language modeling pre-training and fine-tuning. Through comprehensive experiments conducted on BERT, EarlyBERT is shown to reduce training time by approximately 35% to 45%, while maintaining performance at a minimal level of degradation. These results are demonstrated through evaluations on the GLUE and SQuAD benchmarks. GPT-3 is the third iteration of the Generative Pre-trained Transformer (GPT) model series, developed by OpenAI ([73]). The first model, GPT-1, was introduced in 2018 to address the shortage of labeled data by leveraging unlabeled instances. GPT-1, based on an auto-regressive decoder-only transformer with a self-attention mechanism, outperformed state-of-the-art models in various Natural Language Processing (NLP) tasks. GPT-2 was released in 2019 with ten times more parameters and a pre-training corpus ten times larger than GPT-1. Trained on the WebText dataset, GPT-2 set new records in language modeling tasks with zero-shot learning. While it didn't excel in certain NLP subsets, it established compelling baselines that spurred further research on fine-tuning the model. Very large models require complex and sophisticated machines or supercomputers, as announced by Microsoft in 2019,¹ when declaring that the well-known cloud computing platform Azure would have been used together with *gpt-3* language model, whose training requires \$4.6M using a Tesla V100 cloud instance.² Specifically, GPT-3 175B is trained against 499 Billion tokens.³ The figures related to GPT-3 175B model have an even higher magnitude, as the model required 3.14E23 flops of computing for training. A single training runs 355 GPU-years with a resulting training cost of \$4.6M. Based on GPT-3.5 model and presented in November 2022, Open AI ChatGPT (Chat Generative Pre-trained Transformer) [74] exhibits as well as creative and emulation skills, such as the composition of poetry [75] or the emulation of a Linux system [76]. ChatGPT has been trained by using a specific RL algorithm named PPO (Proximal Policy Optimization).⁴ Similarly to GPT-3, the training of ChatGPT involves the use of a transformer architecture. The training process involves several stages, including preprocessing of the input data, training the model

on the pre-processed data, and fine-tuning the model on specific tasks or domains.

Specifically, the self-supervised learning phase is called unsupervised pre-training. The model is trained on a large corpus of text data to learn the underlying patterns and structure of natural language by presenting a sequence of words or tokens. The goal consists of predicting the next word or token in the sequence. This process is known as language modeling.

On Feb 06 2023, Sundar Pichai, CEO of Google and Alphabet, announced the birth of Google Bard (<https://blog.google/intl/en-africa/products/explore-get-answers/an-important-next-step-on-our-ai-journey/>), a large language model chatbot. It is based on the LaMDA family of large language models, and later on PaLM. Bard is trained using a variety of techniques Masked language modeling (MLM), Perplexity minimization, and Generative pre-training (GPT).

Within MLM, the model is given a sentence with some of the words masked out. The model then tries to predict the masked words based on the context of the sentence. Perplexity is a measure of how well a language model can predict the next word in a sequence. In perplexity minimization, the model is trained to minimize its perplexity on a held-out dataset. Finally, in GPT, the model is trained to generate text from scratch. The model is given a prompt and then tries to generate text that continues the prompt.

Bard is based on a Transformer Neural Network Architecture like ChatGPT. For a comparative analysis of the two models, see [77].

C. BigGAN MODELS

Big Generative Adversarial Networks is a DL-based generative model that has been proposed to generate high-resolution and diverse images. BigGAN is a variant of the Generative Adversarial Network (GAN, [78]) framework that was first introduced in 2014. Unlike traditional GANs, which generate low-resolution images, BigGAN can generate images with resolutions up to 512×512 pixels.

The architecture of BigGAN is based on the concept of conditional GANs, where the generator is conditioned on a given class label. The model is trained on a large dataset of images with their corresponding class labels, and it learns to generate new images that are similar to the training set. The training process of BigGAN involves minimizing the distance between the generated images and the real images, while also maximizing the diversity of the generated images.

One of the key features of BigGAN is its ability to generate highly realistic and diverse images. This is achieved by introducing several design choices such as using a hierarchical latent space, using self-attention mechanisms, and incorporating truncation tricks during the generation process. These techniques allow the model to generate images that are not only highly realistic but also contain a wide range of variations within the same class.

BigGAN has been shown to outperform other state-of-the-art generative models on several benchmark datasets.

¹<https://news.microsoft.com/2019/07/22/openai-forms-exclusive-computing-partnership-with-microsoft-to-build-new-azure-ai-supercomputing-technologies>

²<https://lambdalabs.com/blog/demystifying-gpt-3>

³The average token size is 4 characters.

⁴<https://openai.com/blog/openai-baselines-ppo/#ppo>

It has also been used in various applications such as image synthesis, style transfer, and image editing. However, despite its impressive performance, BigGAN still has limitations such as high computational requirements and the need for large amounts of training data. GANs work by training two neural networks against each other. One network, called the generator, is trained to create new data. The other network, called the discriminator, is trained to distinguish between real data and fake data generated by the generator.

The generator and discriminator are trained in a zero-sum game, which means that one network's gain is the other network's loss. The generator tries to generate data that is so realistic that it can fool the discriminator into thinking it is real data. The discriminator tries to become better at distinguishing between real data and fake data.

Over time, the generator and discriminator become better at their respective tasks. The generator becomes better at generating realistic data, and the discriminator becomes better at distinguishing between real and fake data. This process continues until the generator can generate data that is indistinguishable from real data. The BigGAN model is trained using a variety of loss functions, including GAN, Reconstruction, and Perceptual loss. GAN loss is used to train the generator and discriminator. The GAN loss encourages the generator to produce realistic data that fools the discriminator. The reconstruction loss is used to train the encoder and decoder. The reconstruction loss encourages the encoder to learn a useful latent space representation of the data. Finally, the perceptual loss is used to train the generator. The perceptual loss encourages the generator to produce realistic data that is consistent with the perceptual features of real data.

V. SURVEY

This survey has considered 48 articles, which were manually downloaded. Eligible articles were chosen concerning one of the following criteria:

- 1) Introduction to the problem of Green AI. This includes general articles on the topic, discussions regarding trends, and quantitative analyses about the carbon footprint of Red AI systems;
- 2) Quantitative measures regarding the energy requested during the training and inferences phases of the most notable Red AI applications (BERT, GPT, GAN-based models, and PaLM). The mentioned figures are derived from the original articles of different authors. When possible, the section quotes the hardware used by the application;
- 3) Novel approaches to optimize the model training phase, providing *de facto* a transition from Red AI to Green AI.

The text of each remaining article was TF_IDF vectorized, in order to apply the $k - means$ algorithm (note that a more specific discussion and comparison between clustering models is outside of this work). This first step correctly identified 3 clusters renamed as “Cluster 1: Carbon footprint,

sustainability, and Green AI”; “Cluster 2: Bert, GPT and Gan-based models” and finally “Cluster 3: Training optimization”. The clusterization algorithm was applied one more time within only Cluster 1. The same step was then applied to Cluster 2 as well to reveal more details.

A. CLUSTER 1: CARBON FOOTPRINT, SUSTAINABILITY, AND GREEN AI

Cluster 1 is composed of 21 articles about the topic “Carbon footprint, sustainability, and green AI”. The selected papers collectively tackle both challenges and opportunities associated with creating a greener and more sustainable AI and software development. However, the heterogeneity of the components of the cluster ensures a holistic understanding of the relationship between environmental impact, AI, and sustainability. For the sake of clarity, the papers belonging to cluster 1 have been clustered again separately from the others. The result consists of 4 distinct subclusters as per Table 6.

Cluster 1 denotes an evident strive at the global level to emanate principles and guidelines for responsible AI deployment. Furthermore, the need for technological advancement allowing for environmental preservation and social well-being emerges clearly. At the national level, governments are requested to develop policies that appraise the country's contexts and priorities while addressing the environmental AI implications. At the regional and local levels, national policies need to be refined. Additionally, industries promote sustainable AI guidelines and practices within the private sector. The overall result is the emergence of the importance of incorporating sustainability considerations into AI development, deployment, and governance. By identifying environmental impacts, establishing standard units of measurement, and promoting sustainable design practices, policymakers can ensure that AI contributes positively to both societal and environmental well-being, both at a local and global level.

The first sub-cluster is composed of 7 papers and revolves around the concepts of carbon footprint and environmental impact, as the articles explain the challenges and potential solutions for measuring and addressing the ecological consequences of AI usage. Wu et al. [79] underline that, as both AI training and capability lately grew exponentially, the carbon footprint tends to be significantly increasing. From the AI perspective, the authors explain that the carbon footprint is enlarged by the creation, training, and usage of AI models. Additionally, Zhao et al. [80] show that the magnitude of carbon footprint derives also from the energy production operation. AI models computing-intensive training leads to impressive results although at the cost of high energy demand, resulting in a significant carbon footprint. As a result, the environmental cost of electricity production varies, depending on both the region and the energy sources. Energy consumption and carbon footprint are two distinct phenomena that both highlight the need for optimization for reducing the environmental impact [81]. The authors also study both the carbon breakdown of the hardware life cycle

TABLE 5. Case studies in red AI.

Model	Year	Architecture	Training algorithm
BERT	2018	Based on a pile of Transformer encoders	Lottery Ticket Hypothesis (LTH)
BigGAN	2019	Composed of one network, called the generator, is trained to create new data, and a second network, called the discriminator, is trained to distinguish between real data and fake data generated by the generator	Based on GAN, Reconstruction, and Perceptual loss
GPT-3	2020	Based on transformers	In the pre-training phase, the model is trained on a large corpus of diverse text data. After pre-training, the model is fine-tuned on specific tasks or domains to make it more specialized and useful for particular applications.
Bard	2023	Based on Transformers	Based on Masked language modeling (MLM), Perplexity minimization, and Generative pre-training (GPT)

TABLE 6. Carbon footprint, sustainability, and green AI (Sub-clusters).

Topic	Papers	Year of publication
Carbon footprint: causes, implications, numerical measurements, factors that impact on c.f., suggestions to reduce c.f.	7	2021, 2022, 2021, 2020, 2021, 2021, 2022
AI impact on the environment: the necessity for sustainability, regulation state, policymakers perspective, possible improvements.	5	2021, 2021, 2022, 2022, 2022
Energy consumption: AI and energy demand, energy production, and usage, perspectives for reducing energy waste.	4	2019, 2021, 2020, 2021
Efficient code development: objectives, sustainability, perspectives.	5	2021, 2016, 2022, 2022, 2022

and the environmental impact of IT. Anthony et al. [82] and Henderson et al. [83] introduce two tools to measuring carbon footprint and quantifying the impact on the environment, by tracking and showing both energy consumption and carbon emissions. Patterson et al. [84] discuss the veracity of obtained measurements, as finding a truthful estimation is a hard task since underlying costs are often not included in the final calculation. Moreover, estimating energy consumption after the process is completed can give misleading conclusions, thus the difficulty of estimating environmental impact measures retroactively. Patterson et al. [7] present four methodologies that significantly mitigate the energy consumption and carbon emissions of ML workloads compared to conventional alternatives. The authors demonstrate that these methodologies have effectively limited ML’s share of Google’s overall energy consumption to less than 15% over the past three years. Furthermore, they provide a comprehensive explanation for the substantial disparity between published estimates and the actual carbon footprints, revealing a discrepancy ranging from 100 to 100,000 times higher in the former. Based on previous research on energy demand requested by mobile phones adopting ML accelerators, the authors present formulas to calculate the energy consumption being trained to perform a task as proportional to the number of processors employed and the duration of the training run, including the energy requested by a data center, whose energy demand decreased with the implementation of more efficient cloud architectures. Equivalently, the authors can derive the carbon footprint by multiplying the energy value by the carbon intensity of the energy supply. Efforts

are made also to estimate the energy requested by a model to infer decisions. In the conclusions, the authors emphasize the importance of data center providers disclosing specific metrics such as Power Usage Effectiveness (PUE), Carbon-Free Energy (CFE) percentage, and carbon dioxide equivalent (CO₂e) per megawatt hour for each location. Furthermore, ML practitioners should train their models using the most efficient processors available in environmentally friendly cloud data centers. ML researchers are encouraged to develop more efficient ML models, such as by leveraging sparsity and integrating retrieval mechanisms into smaller models, efficiency improvement, carbon footprint reduction, and transparent reporting to foster environmentally conscious practices in the field of machine learning.

The second sub-cluster is composed of 5 papers and revolves around AI sustainability and policy implications. The articles address both direct and indirect environmental impacts of developing and using AI and suggest essential initiatives for policymakers to incorporate AI as a solution to sustainability challenges while minimizing its adverse environmental consequences. Van Wynsberghe et al. [85] propose a definition of sustainable AI, remarking on its dual interpretations: AI for sustainability and sustainability of AI. Specifically, this paper is meant to inspire policymakers, philosophers of AI, and AI developers to connect with the ambient, bearing in mind that AI is inevitably connected to environmental costs. To direct funding towards sustainable methods of AI the author proposes a definition of sustainable AI, which encompasses the entire lifecycle of AI products, from idea generation to implementation and governance. Sustainable AI is not limited to AI applications as it encompasses the broader socio-technical system of AI. The paper characterizes the compatibility of AI development with the sustainable use of environmental resources, economic models, and societal values. In particular, it distinguishes between AI for sustainability and the sustainability of AI per se, particularly focusing on reducing carbon emissions and computing power. By placing sustainable development at its core, Sustainable AI addresses the tensions between AI innovation and equitable resource distribution, inter and intra-generational justice, and the balance between the environment, society, and the economy. Rohde et al. [86] and OECD [87] emphasize the positive opportunities offered by AI for addressing sustainability challenges and propose

the goal of decreasing the negative effects of AI while accelerating its development as a positive influence for the benefit of the planet. Perucica and Andjelkovic [88] provide an overview of existing sustainable AI policy initiatives on multiple levels. Piorkowski et al. [89] discuss the importance of encompassing design, development, deployment, and monitoring of AI systems. The authors propose the assessment of the risks of existing AI models as a means to improve AI regulations and enhance risk management practices.

Sub-cluster 3 is structured into 4 papers collecting Green AI and energy efficiency topics. In this sense, the articles address the environmental impact of developing and using AI from an energy consumption perspective and propose possible improvements to help reduce it. The emergence of Green AI as a focus area highlights the growing recognition of the environmental impact of AI and the need for sustainable practices. Multiple formulas to determine the cost of Red AI have been proposed, considering efficiency and energy consumption as key factors for the calculations. Lacoste et al. [90] use factors such as hardware energy consumption, computation location, emissions associated with the region, and any offsets purchased by the provider to mitigate environmental impact. Lannelongue et al. [91] claim that the environmental impact of an algorithm depends on the energy needed to run it and the pollutants emitted when producing such energy resulting in an evident importance of the used hardware and its energy efficiency. The accuracy improvements in the inference stage require greater computational resources, consequently resulting in greater energy consumption. Therefore, financial and environmental costs tend to grow rapidly. Research and development of new models impact exponentially on the environment as each experiment requires retraining with different model architectures and hyperparameters. Besides exploiting several multiple metrics for efficiency, Strubell et al. [92] measures the energy efficiency of some of the available models in the literature, and suggests reporting the computed training time and sensitivity to hyperparameters as a means to choose both computationally efficient hardware and algorithms when developing. Shaikh et al. [93] consider, among other factors, the geographical position of deployment, thus the preference for more energy-efficient countries.

Sub-cluster 4 is composed of 5 papers and focuses on Green Software Engineering. In detail, the articles address the possible code development opportunities to improve both models' performances and environmental impacts, revolving around the Green software engineering perspective. Green software engineering aims at a development that minimizes the energy consumption, carbon emissions, and overall environmental impact associated with software development and usage. It can be applied by requesting software developers what are the most important requirements for a tool when deployed [94]. In this sense, the energy usage is regarded from multiple perspectives, such as meeting the given requirements, designing, coding, and debugging.

Georgiou et al. [95] tackle the issue related to the balance of energy and run-time efficiency, discussing the need for a trade-off between run-time performance and energy consumption. The authors claim that it is not possible to identify a framework performing optimally for each task. As a result, users should appraise both energy and performance requirements when designing an AI model. Verdecchia et al. [96] recommend a data-centric approach where data is effectively manipulated before further operations, as it strongly enhances AI efficiency at a low cost. Although AI implies inevitably environmental costs, it can be helpful in mitigating issues occurring in other areas. For example, Pachot and Patissier [97] underline the existence of a contradiction in a technology that requires a great deal of energy being tasked with addressing ecological concerns. The authors provide a comprehensive view of sectors that employ AI-powered solutions for environmental preservation. This includes smart city management, the energy sector, agriculture, disaster prediction and adaptation to climate change, ecosystem protection, transportation, and economics. Additionally, Vinuesa et al. [98] view AI as a challenge. Depending on how it is used, it might either help or contrast the reaching of sustainability goals set by the 2030 Agenda for Sustainable Development.

TABLE 7. A = Environmental impact and sustainability, B = Carbon footprint, C = Energy efficiency, D = Green software engineering, E = Green AI, F = Policy considerations.

	A	B	C	D	E	F
[79]	X	X				
[80]		X				
[81]		X	X			
[82]		X	X			
[83]		X	X			
[84]		X				
[7]		X	X			
[85]	X					X
[86]	X					X
[87]	X					X
[88]	X					X
[89]				X		X
[90]		X				
[91]		X	X			
[92]	X	X	X			X
[93]		X	X			
[94]				X		
[95]			X	X		
[96]			X			
[97]	X		X			
[98]	X					X

Table 7 provides insights into the distribution of the main themes that characterize the cluster. The chosen tags include environmental impact and sustainability (A), carbon footprint (B), energy efficiency (C), green software engineering (D), green AI (E), and policy considerations (F). Carbon Footprint (B) and Energy Efficiency (C) are the most prevalent themes across the papers, suggesting that a significant portion of the research focuses on the carbon emissions and energy consumption aspects from multiple points of view. Vice versa, Green Software Engineering (D) and Green AI (E) appear in a few papers, indicating a growing interest in

the highlighted themes and underlying the recentness of the emergence of the need for more green development and more regulations aimed at sustainable practices.

B. CLUSTER 2: BERT, GPT, PaLM AND BIGGAN MODELS

As per Table 8, cluster 2 is composed of 15 articles that have been further decomposed in 3 subclusters referring to the discussion of the architectures models such as BERT (including derived models), GPT, PaLM and BigGAN.

TABLE 8. BERT, GPT, PaLM and bigGAN models (Sub-clusters).

Topic	Papers
BERT (and derived models) [99], [100], [101], [68], [102], [103], [104], [105], [106], [107], [108]	11
GPT [109], [110]	2
BigGAN [111]	1
PaLM [112]	1

After pre-training on BookCorpus,⁵ BERT can be fine-tuned on specific downstream tasks, such as sentiment analysis or named entity recognition, by training on task-specific labelled data. According to Devlin et al. [99], the BERT SQuAD model (Stanford Question Answering Dataset, a reading comprehension dataset) can be trained in around 30 minutes on a single Cloud TPU to achieve a Dev F1 score of 91.0%. The authors report that the two training phases add up to a total of 40 epochs and take over 80 hours to complete on a 64 TPUv3 chip.

Several studies have estimated the energy consumption of training large language models such as BERT. Although the considered NLP models exploited for the study are not recent, the authors firstly estimate total power p_t required at a given instance during training and secondly, they convert power to estimated CO_2 emissions. By using these variables, the paper reviews the cost of training different NLP models. The authors stress the fact that future scientific research should report training time and sensitivity to model hyperparameters. In particular, Strubell et al., recommend also ethical conduct of research, i.e. prioritizing hardware and algorithms that are computationally efficient. Schwartz et al. estimated that training BERT on a single GPU could emit as much carbon dioxide as a transatlantic flight. Efforts are being made to reduce the energy consumption of training large language models. For example, researchers are exploring techniques such as model distillation, which involves training a smaller model to mimic the behavior of a larger pre-trained model.

Training is a task that can be accomplished in different ways. For example, in [101], the authors discuss a scenario where a BERT is trained within 2 weeks on an academic-size cluster of GPUs by using efficient optimizations (see also [68], which reviews the fine-tuning process of BERT and its performance on several NLP tasks and [102], providing a detailed guide on how to fine-tune BERT for text classification tasks).

SpanBERT [103] is a language representation model that was pre-trained on large amounts of text data. Differently from BERT and GPT, which were trained to predict the next word in a sequence, SpanBERT predicts spans of text within a sentence. In the masked language modeling (MLM) step, a random portion of each input sequence is replaced with a MASK token, then the model is trained to predict the original words in the masked positions. In the span boundary objective (SBO) step, the model is trained to predict the boundaries of spans of text within the input sequence. This step was achieved by employing a specific objective function that awards the model when considering all possible spans. According to the authors, the pre-training phase used a large amount of text data from the English Wikipedia and BookCorpus, and was done on 32 Volta V100 GPUs, taking 15 days to complete. Information about the estimate of the power consumption and the carbon footprint was not available.

BioBERT [104] is a pre-trained language model for biomedical text mining tasks. BioBERT is based on the BERT architecture, which is a transformer-based language model originally pre-trained on a large corpus of general text data. However, BioBERT was further pre-trained on a large corpus of biomedical text data to better capture domain-specific information and improve performance on biomedical text mining tasks. To pre-train BioBERT, the authors used a large corpus of biomedical text data from various sources, including PubMed abstracts and full-text articles, as well as clinical notes from the MIMIC-III dataset. The pre-training process involved both masked language modeling (MLM) and next sentence prediction (NSP) tasks, similar to the original BERT model. The cost of training BioBERT is estimated in [105], where the authors propose a more parsimonious architecture for Domain adaptation of Pretrained Language Models (PTLMs) is typically achieved by unsupervised pretraining on target-domain text. BERT is trained by using 8 NVIDIA v100 GPUs (32GB), requiring a power of 1505W, an execution time equal to 552 hours, and an overall carbon footprint of 1252 lbs of CO_2 .

According to the authors, the ALBERT model [106] was trained on multiple accelerators, including a cluster of 64 TPUv3 chips and multiple GPU clusters consisting of hundreds of GPUs. Regarding the training time, the authors mentioned that they trained the ALBERT-xxlarge model on a cluster of 64 TPUv3 chips for about 3 days. They also reported that they trained the smaller ALBERT models on a GPU cluster for several days. As for the energy consumption during training, the paper did not provide an exact figure for this metric. However, the authors did mention that they used mixed-precision training and gradient accumulation to reduce the memory footprint and energy consumption during training. They also used several other optimization techniques to speed up training, including dynamic down-sampling, and alternating optimization.

ROBERTa [107] uses the same pre-training procedure as per BERT, which was trained on multiple accelerators,

⁵<https://paperswithcode.com/dataset/bookcorpus>

including TPU v3 and v3-8, as well as GPU clusters. RoBERTa was trained for several epochs on the BooksCorpus and English Wikipedia, using a batch size of 8,192 tokens per GPU. The training time and energy consumption of RoBERTa would depend on various factors, such as the specific hardware used, the number of epochs, and the batch size. Without specific information about these factors, it is difficult to estimate the training time and energy consumption accurately.

In [108], the authors mention that they used the pre-training procedure of BERT, which was trained on multiple accelerators, including TPU v3 and v3-8, as well as GPU clusters. Specifically, block-structured pruning was deployed to reduce the computational and memory cost of the transformer model. The authors applied pruning to both the weights and activations of the model and achieved a significant reduction in the number of parameters without a significant loss in accuracy.

The training of the original GPT model, which had 117 million parameters, was performed on 8 NVIDIA V100 GPUs for several days [109]. The training data consisted of a web crawl dataset containing 40 GB of text. The larger GPT-2 model, which had up to 1.5 billion parameters, was trained on a cluster of 512 TPU v3 chips for several days. The training data consisted of a much larger web crawl dataset and several other text sources [110]. As for the energy consumption of GPT training, there have been some concerns about the environmental impact of training large language models like GPT-2.

The training process of BigGAN [111] involves two components: the generator and the discriminator. The generator is a deep neural network that takes as input a random vector sampled from a high-dimensional latent space and produces an image. The discriminator is another deep neural network that takes as input an image and outputs a scalar value that indicates how realistic the image is. During training, the generator tries to produce images that can fool the discriminator into thinking they are real, while the discriminator tries to distinguish between the real images and the fake images produced by the generator. BigGAN has been trained on large-scale datasets such as ImageNet and CIFAR-10, which contain millions of images with corresponding class labels. The generator of BigGAN is conditioned on the class label of the image, which means that it learns to generate images that are specific to a particular class. One of the key features of BigGAN is its use of a progressive training strategy, which involves training the model on low-resolution images first and gradually increasing the resolution over time. This allows the model to learn low-level features such as edges and textures before moving on to higher-level features such as shapes and objects. In addition, BigGAN incorporates a number of design choices such as using a hierarchical latent space, using self-attention mechanisms, and incorporating truncation tricks during the generation process. These techniques allow the model to generate images that are not only highly realistic but also contain

a wide range of variations within the same class. One (512px) BigGAN experiment is equivalent to a trans-Atlantic roundtrip (1 to 2t of CO₂). Training a deep learning model like BigGAN requires a significant amount of energy, as it involves processing large amounts of data and performing many complex computations. The exact amount of energy required to train BigGAN can vary depending on the specific configuration of the model, the size of the training dataset, and the computing hardware used. Strubell et al. estimated that training a large-scale GAN model similar to BigGAN for 1 hour could consume as much as 284 kWh of electricity, which is equivalent to the energy consumption of an average American household in a week. This estimate was based on the assumption that the model was trained on a single GPU and that the electricity was generated using a mix of coal, natural gas, and nuclear power. It is worth noting that there have been efforts to develop more energy-efficient deep learning algorithms and hardware architectures that can reduce the energy consumption of training models like BigGAN. For example, recent research has explored the use of sparsity-inducing techniques, which can reduce the number of computations required during training, and the use of specialized hardware such as tensor processing units (TPUs), which can perform deep learning computations more efficiently than traditional CPUs or GPUs.

PaLM (Pre-training with a Large-scale Memory [112]), is one of the largest and most powerful LLMs in the world, with 540 billion parameters. PaLM was trained on a massive dataset of text and code, and it can perform a wide range of tasks. PaLM-540B was trained on 6144 TPU v4 chips for 1200 hours and 3072 TPU v4 chips for 336 hours (this included external factors, such as downtime and repeated steps).

Table 9 summarises the training cost for the most relevant applications.

C. TRAINING OPTIMISATION

This cluster consists of 12 articles, which are not decomposed further because of the homogeneity of the result. The discussed topic concerns the optimization of large models through different techniques to curb the training and inference time. See Table 10 for a summary of the different approaches.

Distributed training [121] involves training a machine learning model using multiple computational resources, such as multiple GPUs or multiple machines. It aims to accelerate the training process by dividing the workload among different resources and exchanging information during the training process. The interesting point is that this type of training allows for parallel processing across multiple GPUs or machines, significantly reducing the training time. It also enables efficient utilization of computational resources, increasing scalability and enabling the training of larger models. A Possible drawback is that it requires additional setup and coordination among different resources. Synchronization and communication overhead between the resources can introduce latency and increase complexity.

TABLE 9. Training cost of the most relevant models.

Application	Training Data	Hardware	Energy requested (training)	Carbon footprint
BERT	Bookcorpus	64 TPUv3 chips	between 300 and 700 kWh (kilowatt-hours) of energy	280 to 630 kgCO ₂
GPT3	Web crawl dataset containing 40 GB of text	Tesla V100 GPU cluster	50MWh	Lifetime emissions of five cars
spanBERT	English Wikipedia and BookCorpus	32 Volta V100 GPUs	N/A	N/A
BioBERT v1.1	PubMed abstracts and full-text articles and clinical notes from the MIMIC-III dataset	base 8 NVIDIA v100 GPUs (32GB)	1505W 552 hours	1252 lbs CO ₂
PaLM	BooksCorpus, the English portion of the Wikipedia corpus, Common Crawl, and other large-scale datasets	6144 TPU v4 chips for 1200 hours and 3072 TPU v4 chips for 336 hours	378.5W measured system power per TPU v4 chip	271.43 tCO ₂ e
ROBERTa	BooksCorpus and English Wikipedia	64 TPUv3 chips	N/A	N/A
BigGAN	large-scale datasets such as ImageNet and CIFAR-10	Google TPU v3 Pod, with the number of cores proportional to the resolution	1-hour training, could consume as much as 284 kWh of electricity	Equivalent to a trans-Atlantic roundtrip (≈ 1 to 2t of CO ₂)

Transfer learning [115] involves leveraging the knowledge gained from training one model on a specific task and applying it to a different but related task. Instead of training a model from scratch, transfer learning utilizes pre-trained models as a starting point and fine-tunes them on the new task. It can help in situations where labeled training data is limited or expensive. Typically, transfer learning leverages pre-trained models and knowledge from one task to another, reducing the need for extensive training on new data. It can improve model performance, especially when labeled training data is limited or expensive. However, this may not be effective if the pre-trained model is not well-suited to the new task or if the data distributions between the pre-training and target tasks differ significantly.

Progressive learning [114] refers to a learning paradigm where a model is incrementally trained on new data while retaining previously learned knowledge. It allows the model to adapt to new information without completely forgetting the knowledge gained in earlier stages. Progressive learning is particularly useful in scenarios where the distribution of data changes over time or when continuous learning is required. It enables incremental learning on new data while retaining previously learned knowledge, allowing the model to adapt to changing data distributions. Progressive learning can improve model performance in scenarios where data changes over time or continuous learning is required, but it requires careful management of knowledge retention and integration, as catastrophic forgetting of previously learned knowledge can occur. It may also lead to slower overall training due to the incremental nature of learning.

Mixed precision training [122] involves using different numerical precision (e.g., using a combination of lower and higher precision formats) during the training process. This technique exploits the fact that some parts of the model can tolerate lower precision without significant loss of accuracy, enabling faster training and reduced memory consumption. In this case, the process utilizes lower precision data types

for certain model components, speeding up and reducing memory consumption without significant accuracy loss. This approach enables faster training on specialized hardware and can be beneficial for large-scale models. As a drawback, it must be remarked that this type of training requires careful balancing and management of precision levels to maintain model accuracy. In the worst of cases, it may introduce numerical instability issues if not implemented properly.

In the context of machine learning models, sparsity refers to a property where a significant portion of the model's weights or activations are zero. Sparse models can have computational and memory advantages by reducing the number of computations and the memory footprint. Techniques such as weight pruning or regularisation methods can be used to induce sparsity in a model [118]. Sparse models have computational and memory advantages, reducing the number of computations and memory footprints. They can improve inference speed and enable efficient deployment on resource-constrained devices. On the other hand, inducing sparsity in models may require additional techniques such as weight pruning or regularisation, which can add complexity to the training process. Sparse models may also require specialized hardware or software support for efficient implementation.

Model compression encompasses a set of techniques used to reduce the size of a trained model without significantly sacrificing its performance. It involves methods like weight quantization, pruning, knowledge distillation, and compact architecture design [117]. Model compression is crucial for deploying models on resource-constrained devices or reducing storage and bandwidth requirements. Typically, model compression techniques reduce the size of trained models, enabling efficient storage and deployment on resource-constrained devices. They can also reduce memory and bandwidth requirements, improving inference speed. However, these techniques may introduce a trade-off between model size reduction and performance. In some cases, compressing the model too much can lead to a significant

TABLE 10. Optimisation models evaluation.

Class	Technique	Year of publication	Evaluation
Hardware	Specialized hardware [113]	2019	Training a large language model can be accelerated by using TPUs
Learning	Progressive learning [114]	2017	It enhances model performance in dynamic or continuous learning settings. However, managing knowledge retention is crucial to prevent catastrophic forgetting, and the incremental nature of learning can result in slower overall training.
Learning	Transfer learning [115]	2023	It improves model performance in new tasks, particularly when labeled training data is scarce or costly. However, its effectiveness depends on the suitability of the pre-trained model and the similarity of data distributions between the pre-training and target tasks.
Heuristics	Early stopping [116]	2020	It strikes a balance between model complexity and generalization. However, as it relies on heuristics, it may not always produce the best-performing model. Monitoring training progress and validation performance is crucial to avoid premature stopping or suboptimal outcomes.
Heuristics	Model compression [117]	2020	It reduces the model size for efficient storage and deployment on resource-constrained devices, improving memory usage and inference speed. However, there is a trade-off between size reduction and performance, as excessive compression can result in accuracy loss. Complex compression methods may introduce computational overhead during training and inference.
Heuristics	Sparsity models [118]	2021	Offer computational and memory benefits by reducing computations and memory usage, leading to improved inference speed and efficient deployment on resource-constrained devices. However, inducing sparsity may require additional techniques and specialized support, adding complexity to the training process.
Heuristics	Knowledge distillation [56]	2021	It reduces memory requirements, improves inference speed, and enables deployment on resource-limited devices. It generalizes well to new tasks, learning from the pre-trained model's rich knowledge. However, it may result in a slight performance degradation compared to the original model, requiring additional computational resources and training time for both teacher and student models, making it computationally expensive.
Heuristics	Data augmentation [119]	2021	Employed to expand training data, improving model performance and mitigating overfitting. They prove beneficial in scenarios with limited labeled data, but improper implementation can introduce artificial patterns or noise, impacting the model's generalization. Additionally, certain augmentation methods may raise computational expenses during training.
Training	Parallel training [120]	2020	It concurrently trains multiple model instances, notably expediting training and optimizing resource use for intricate models. Yet, this method introduces synchronization issues, potentially elevating latency due to communication overhead. Critically, certain models/tasks lack inherent parallelizability, curbing potential speed enhancements.
Training	Distributed training [121]	2020	It allows for parallel processing across multiple GPUs or machines, reducing training time and efficiently utilizing computational resources. However, it requires additional setup and coordination, introducing synchronization and communication overhead that may result in latency and increased complexity.
Training	Mixed precision training [122]	2022	It results in faster training and reduced memory consumption with minimal accuracy loss. It is particularly advantageous for large-scale models and specialized hardware. However, careful precision level management is necessary to maintain accuracy, as improper implementation can lead to numerical instability issues in extreme cases.
Training	Pre-training BERT on smaller data [123]	2023	It accelerates training time with good performance and facilitates faster experimentation and model development on limited resources. However, pre-training on smaller data may restrict the model's capacity to capture diverse language patterns and may not generalize well to various downstream tasks, particularly if the pre-training data does not adequately represent the target task domain.

drop in accuracy. Additionally, complex compression techniques may add computational overhead during training and inference.

Knowledge distillation [56] involves training a smaller, more lightweight model to mimic the behavior of a larger, more complex model like BERT. This approach can significantly reduce the training time and memory requirements of the model. It can reduce memory requirements, improve inference speed, enable deployment on resource-limited devices, and can generalize well to new tasks or domains, as the distilled model learns from the rich knowledge of the pre-trained model. However, knowledge distillation may not fully capture all the nuances and complexities of the original

model, resulting in a slight degradation in performance compared to the larger model. Furthermore, this technique requires additional computational resources and training time to train both the teacher (pre-trained model) and student (distilled model), making it more computationally expensive.

Pre-training BERT on smaller data can significantly reduce the training time while still achieving good performance [123]. It has to be noticed that this approach can reduce training time while still achieving good performance. It allows for faster experimentation and model development on limited resources. On the other hand, pre-training BERT on smaller data may limit the model's ability to capture a wide range of language patterns and nuances. It may not

generalize well to diverse downstream tasks, especially if the pre-training data does not adequately represent the target task domain.

The use of specialized hardware such as TPUs and GPUs can significantly reduce the training time of deep learning models. For example, [113] shows that training a large language model can be accelerated by using TPUs.

Increasing the amount of training data can improve the performance of the model [119] and reduce the amount of training time required. This can be achieved through data augmentation techniques such as back-translation and paraphrasing. Specifically, Data augmentation techniques increase the amount of training data, which can enhance model performance and reduce overfitting. This approach can be particularly useful when labeled training data is scarce, although artificial patterns or noise may occur. The drawback consists, in this case, of potentially affecting the model's ability to generalize to real-world data. Some augmentation techniques may also increase the computational cost during training.

Early stopping is a technique that involves stopping the training process before the model has converged to the optimal solution. This can help reduce the training time and prevent overfitting. In [116] the authors discussed that early stopping can be an effective technique for reducing the training time of deep learning models. The method is rather interesting as it reduces training time by terminating the training process before convergence to the optimal solution. In particular, it helps in finding a good trade-off between model complexity and generalization. However, since it relies on heuristics to determine the stopping point, it may not always lead to the best-performing model. Moreover, it requires careful monitoring of training progress and validation performance to avoid premature stopping or suboptimal results. Finally, parallel training [120] involves training multiple instances of the model in parallel, which can significantly reduce the training time. Furthermore, it enables efficient utilization of computational resources, allowing for the training of larger and more complex models, and provides scalability, as parallel training can be easily extended to utilize additional GPUs or machines. As a drawback, it introduces synchronization and communication overhead between instances, which can increase latency and complexity. As a more serious drawback, it must be taken into account that some models or tasks may not be inherently parallelizable, limiting the potential speedup achievable through parallel training.

VI. CONCLUSION AND FUTURE WORK

AI supports a green society in different ways. AI technologies such as remote sensing, drones, and satellite imagery provide farmers with detailed insights about their fields. This information helps optimize irrigation, fertilization, and pest control, resulting in higher crop yields and reduced resource wastage; AI-powered traffic management systems analyze real-time data from sensors, cameras, and GPS devices

to optimize traffic flow. AI-driven building management systems analyze occupancy patterns, weather forecasts, and other data to predict energy demand for heating, cooling, lighting, and other building functions. However, there are scenarios where AI can negatively affect the solutions it provides because of its excessive energy demand, due to insufficient hardware architectures, poor choice of data structures to represent very large datasets or inefficient and expensive training algorithms. As a paradox, Red AI can cause a more serious problem than the ones it solves. Following this direction, this article has critically reviewed the different causes of Red AI in terms of architectures, data structures, and algorithms aiming to reduce the computational complexity occurring during the training of a model. The principles of the most energy-demanding DL applications are reviewed as a foundational pillar of a survey that takes into account a set of articles introducing the topic of Green AI. A second cluster of articles discussed the hardware used and the energy demand of the most computationally intensive DL models. Finally, a third group of contributions related to the latest advancements with respect of the training optimization of large models has been debated. Green AI often involves finding a balance between energy efficiency and performance. However, it is here to stay, and it will soon become a pillar of our society.

Firstly, there is a critical need from an architectural point of view. Exploring hardware innovations for reduced energy consumption and the consequent carbon emissions is of paramount importance. The reduction of consumed energy can be achieved by creating more optimized data structures, algorithms, and models, which can reduce the computational complexity and save a significant amount of electrical power, thereby reducing the carbon footprint. Integration of renewable energy sources into AI infrastructure should be a central theme, with an emphasis on minimizing energy consumption in smart buildings. While it is often advised to power everything using alternative resources to gas, carbon, and the like, the impact of such actions is not always properly calculated. For instance, preferring an electric vehicle to a petrol vehicle is environmentally friendly from a direct pollution creation perspective. However, the environmental impact depends on how the electricity that powers these devices is produced. If a strongly polluting method of energy production is preferred, even electric cars can be indirectly polluting. Therefore, it is essential to thoroughly assess the sustainability of seemingly sustainable choices.

From an energetic point of view, there is a need for in-depth examination. Certainly, producing energy can be less harmful to the environment, but not all the resources used in the process are sustainable. Therefore, using an electric and optimized device that consumes energy produced in an environmentally harmful way can still result in a large carbon footprint. Such production methods should be analyzed and regulated in further research. Furthermore, the application of AI in environmental monitoring and resource management

must be thoroughly explored. The limitations of this work can be regarded as the lack of a debate concerning the governmental policies (in the form of norms, practices and directives) to reduce the presence of Red AI in favor of Green AI. In this sense, future work will review the issues related to the negotiation between sustainable AI and the escalating energy demands of AI models, as seeking optimal performance involves finding a balance between performance requirements and environmental considerations. Specifically, the points targeted by future work will include i) how to prioritize the optimization of algorithms and models to enhance their efficiency, ii) review energy-efficient hardware solutions, iii) delve into techniques such as quantization and model compression to reduce the size of models, leading to lower computational requirements and, consequently, reduced energy consumption, iv) assess techniques based on scaling up or down the computational resources based on the current demand, v) review the environmental impact at each stage the entire life-cycle of AI models and seek ways to minimize the carbon footprint, vi) study methods to power AI infrastructure with renewable energy sources to mitigate the environmental impact, vii) discuss the possibility of complying with regulations related to energy consumption and environmental impact, viii) review incentives or recognition programs for AI developers and organizations that adopt sustainable practices and finally, ix) describe continuous monitoring policies of energy consumption and performance metrics

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
- [2] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–25, Jul. 2021.
- [3] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, Mar. 2022.
- [4] F. Yu, X. Xiu, and Y. Li, "A survey on deep transfer learning and beyond," *Mathematics*, vol. 10, no. 19, p. 3619, Oct. 2022.
- [5] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106622.
- [6] R. Schwartz, J. Dodge, N. Smith, and O. Etzioni, "Green AI," *Commun. ACM*, vol. 63, no. 12, pp. 54–63, Dec. 2020.
- [7] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L. Munguia, D. Rothchild, D. R. So, M. Texier, and J. Dean, "The carbon footprint of machine learning training will plateau, then shrink," *Computer*, vol. 55, no. 7, pp. 18–28, Jul. 2022.
- [8] J. von Neumann, "First draft of a report on the EDVAC," *IEEE Ann. Hist. Comput.*, vol. 15, no. 4, pp. 27–75, Jan. 1993.
- [9] B. A. Kahle and W. D. Hillis, "The connection machine model CM-1 architecture," *IEEE Trans. Syst. Man, Cybern.*, vol. 19, no. 4, pp. 707–713, Jul. 1989.
- [10] M. Sánchez-Rico, N. Hoertel, and J. M. Alvarado, "Combination of cluster analysis with dimensionality reduction techniques for pattern recognition studies in healthcare data: Comparing PCA, t-SNE and UMAP," *PsyArXiv*, Jan. 2023. [Online]. Available: <https://osf.io/preprints/psyarxiv/zxvf2>
- [11] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AICHE J.*, vol. 37, no. 2, pp. 233–243, Feb. 1991.
- [12] A. Hasegawa, S.-C. Lo, J.-S. Lin, M. Freedman, and S. Mun, "A shift-invariant neural network for the lung field segmentation in chest radiography," *VLSI Signal Process.*, vol. 18, pp. 241–250, Apr. 1998.
- [13] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.
- [14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [15] P. Domingos, *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. New York, NY, USA: Basic Books, 2015.
- [16] M. Minsky and S. A. Papert, *Perceptrons, Reissue 1988 Expanded, Ed., With a New Foreword by Léon Bottou: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 2017.
- [17] D. R. Williams and Y. Tang, "Impact of office productivity cloud computing on energy consumption and greenhouse gas emissions," *Environ. Sci. Technol.*, vol. 47, no. 9, pp. 4333–4340, May 2013.
- [18] A. Vishwanath, F. Jalali, K. Hinton, T. Alpcan, R. W. A. Ayre, and R. S. Tucker, "Energy consumption comparison of interactive cloud-based and local applications," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 4, pp. 616–626, Apr. 2015.
- [19] E. Barbierato, D. Manini, and M. Gribaudo, "A multiformalism-based model for performance evaluation of green data centres," *Electronics*, vol. 12, no. 10, p. 2169, May 2023.
- [20] J. Xu, W. Zhou, Z. Fu, H. Zhou, and L. Li, "A survey on green deep learning," 2021, *arXiv:2111.05193*.
- [21] R. Verdecchia, J. Sallou, and L. Cruz, "A systematic review of green AI," 2023, *arXiv:2301.11047*.
- [22] S. Salehi and A. Schmeink, "Data-centric green artificial intelligence: A survey," *IEEE Trans. Artif. Intell.*, vol. 1, no. 1, pp. 1–18, Sep. 2023.
- [23] N. Siegmund, J. Dorn, M. Weber, C. Kaltenecker, and S. Apel, "Green configuration: Can artificial intelligence help reduce energy consumption of configurable software systems?" *Computer*, vol. 55, no. 3, pp. 74–81, Mar. 2022.
- [24] M. Gutiérrez, M. Á. Moraga, F. García, and C. Calero, "Green-IN machine learning at a glance," *Computer*, vol. 56, no. 6, pp. 35–43, Jun. 2023.
- [25] N. Talati, R. Ben-Hur, N. Wald, A. Haj-Ali, J. Reuben, and S. Kvatinsky, "mMPU—A real processing-in-memory architecture to combat the von Neumann bottleneck," in *Applications of Emerging Memory Technology*. New York, NY, USA: Springer, 2020.
- [26] R. Pawson, "The myth of the Harvard architecture," *IEEE Ann. Hist. Comput.*, vol. 44, no. 3, pp. 59–69, Jul. 2022.
- [27] W. S. McCulloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.
- [28] J. S. Vetter, *Contemporary High Performance Computing: From Petascale Toward Exascale*. Boca Raton, FL, USA: CRC Press, 2013.
- [29] L. Wang and S. U. Khan, "Review of performance metrics for green data centers: A taxonomy study," *J. Supercomput.*, vol. 63, no. 3, pp. 639–656, Mar. 2013.
- [30] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario, "The future of quantum computing with superconducting qubits," *J. Appl. Phys.*, vol. 132, no. 16, Oct. 2022, Art. no. 160902.
- [31] N. Elsayed, A. S. Maida, and M. Bayoumi, "A review of quantum computer energy efficiency," in *Proc. IEEE Green Technol. Conf. (GreenTech)*, Apr. 2019, pp. 1–3.
- [32] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, Dec. 1997.
- [33] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.
- [34] T. Kohonen, "Self-organized formation of topologically correct feature maps," in *Biological Cybernetics*, vol. 43. Cham, Switzerland: Springer, 1982, pp. 59–69.
- [35] F. Rosenblatt, *Perceptron Simulation Experiments*, vol. 48, no. 3. New York, NY, USA: IEEE, 1960.
- [36] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.
- [37] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, Jun. 1991.
- [38] A. K. Tyagi and A. Abraham, "Recurrent neural networks: Concepts and applications," *Tech. Rep.*, 2022.

- [39] N. Suresh, N. C. A. Kumar, S. Subramanian, and G. Srinivasa, "Memory augmented recurrent neural networks for de-novo drug design," *PLoS ONE*, vol. 17, no. 6, Jun. 2022, Art. no. e0269461.
- [40] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.
- [41] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing machines," 2014, *arXiv:1410.5401*.
- [42] W. Zaremba and I. Sutskever, "Reinforcement learning neural Turing machines—revised," 2015, *arXiv:1505.00521*.
- [43] R. B. Greve, E. J. Jacobsen, and S. Risi, "Evolving neural Turing machines for reward-based learning," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2016, pp. 117–124.
- [44] G. Yang, "Lie access neural Turing machine," 2016, *arXiv:1602.08671*.
- [45] C. Gulcehre, S. Chandar, K. Cho, and Y. Bengio, "Dynamic neural Turing machine with soft and hard addressing schemes," 2016, *arXiv:1607.00036*.
- [46] K. Kurach, M. Andrychowicz, and I. Sutskever, "Neural random-access machines," 2015, *arXiv:1511.06392*.
- [47] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neuromorphic chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.
- [48] H. Markram, "The blue brain project," *Nature Rev. Neurosci.*, vol. 7, no. 2, pp. 153–160, Feb. 2006.
- [49] S. Lanthaler and A. M. Stuart, "The curse of dimensionality in operator learning," 2023, *arXiv:2306.15924*.
- [50] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev. Soc. Ind. Appl. Math.*, vol. 51, no. 3, pp. 455–500, Aug. 2009.
- [51] S. Rabanser, O. Shchur, and S. Günnemann, "Introduction to tensor decompositions and their applications in machine learning," 2017, *arXiv:1711.10781*.
- [52] Y. Ji, Q. Wang, X. Li, and J. Liu, "A survey on tensor techniques and applications in machine learning," *IEEE Access*, vol. 7, pp. 162950–162990, 2019.
- [53] A. Bernardi, J. Brachat, P. Comon, and B. Mourrain, "General tensor decomposition, moment matrices and applications," *J. Symbolic Comput.*, vol. 52, pp. 51–71, May 2013.
- [54] A. G. M. Lewis, J. Beall, M. Ganahl, M. Hauru, S. B. Mallick, and G. Vidal, "Large scale distributed linear algebra with tensor processing units," 2021, *arXiv:2112.09017*.
- [55] R. Mayer and H.-A. Jacobsen, "Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–37, Jan. 2021.
- [56] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.
- [57] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [58] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
- [59] Y. LeCun, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
- [60] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," 2016, *arXiv:1603.05279*.
- [61] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, 2015, pp. 3123–3131.
- [62] V. Sindhwani, T. N. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1664–1672.
- [63] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014, *arXiv:1404.5997*.
- [64] T. Wiedmann and J. Minx, "A definition of 'carbon footprint,'" *Ecol. Econ. Res. trends*, vol. 1, pp. 1–11, Mar. 2008.
- [65] K. S. Jones, *Natural Language Processing: A Historical Review*. Dordrecht, The Netherlands: Springer, 1994, pp. 3–16.
- [66] P. Johri, S. K. Khatri, A. T. Al-Taani, M. Sabharwal, S. Suvanov, and A. Kumar, "Natural language processing: History, evolution, application, and future work," in *Proc. 3rd Int. Conf. Comput. Informat. Netw. (ICCN)*. Cham, Switzerland: Springer, 2021, pp. 365–375.
- [67] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," 2019, *arXiv:1906.08237*.
- [68] C. Raffel, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 5485–5551, 2020.
- [69] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in BERTology: What we know about how BERT works," *Trans. Assoc. Comput. Linguist.*, vol. 8, pp. 842–866, Jan. 2021.
- [70] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, Jan. 2022.
- [71] X. Chen, Y. Cheng, S. Wang, Z. Gan, Z. Wang, and J. Liu, "Early-BERT: Efficient BERT training via early-bird lottery tickets," 2020, *arXiv:2101.00063*.
- [72] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018, *arXiv:1803.03635*.
- [73] M. Zong and B. Krishnamachari, "A survey on GPT-3," 2022, *arXiv:2212.00857*.
- [74] T. Wu, S. He, J. Liu, S. Sun, K. Liu, Q.-L. Han, and Y. Tang, "A brief overview of ChatGPT: The history, status quo and potential future development," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 5, pp. 1122–1136, May 2023.
- [75] A. R. Kirmani, "Artificial intelligence-enabled science poetry," *ACS Energy Lett.*, vol. 8, no. 1, pp. 574–576, Jan. 2023.
- [76] N. M. S. Surameery and M. Y. Shakor, "Use chat GPT to solve programming bugs," *Int. J. Inf. Technol. Comput. Eng.*, no. 31, pp. 17–22, Jan. 2023.
- [77] I. Ahmed, M. Kajol, U. Hasan, P. P. Datta, A. Roy, and M. R. Reza, "ChatGPT vs. Bard: A comparative study," Authorea, New York, NY, USA, 2023.
- [78] I. Goodfellow, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [79] C.-J. Wu, "Sustainable AI: Environmental implications, challenges and opportunities," 2021, *arXiv:2111.00364*.
- [80] D. Zhao, N. C. Frey, J. McDonald, M. Hubbell, D. Bestor, M. Jones, A. Prout, V. Gadepally, and S. Samsi, "A Green(er) world for A.I.," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Lyon, France, May 2022, pp. 742–750.
- [81] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H. S. Lee, G. Wei, D. Brooks, and C. Wu, "Chasing carbon: The elusive environmental footprint of computing," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2021, pp. 854–867.
- [82] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: Tracking and predicting the carbon footprint of training deep learning models," 2020, *arXiv:2007.03051*.
- [83] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 10039–10081, 2020.
- [84] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," 2021, *arXiv:2104.10350*.
- [85] A. Van Wynsberghe, "Sustainable AI: AI for sustainability and the sustainability of AI," *AI Ethics*, vol. 1, no. 3, pp. 213–218, 2021.
- [86] F. Rohde, M. Gossen, J. Wagner, and T. Santarius, "Sustainability challenges of artificial intelligence and policy implications," *Ökologisches Wirtschaften Fachzeitschrift*, vol. 36, no. 1, pp. 36–40, Feb. 2021.
- [87] *Measuring the Environmental Impacts of Artificial Intelligence Compute and Applications*, OECD, Paris, France, 2022.
- [88] N. Perucica and K. Andjelkovic, "Is the future of AI sustainable? A case study of the European union," *Transforming Government, People, Process Policy*, vol. 16, no. 3, pp. 347–358, Jul. 2022.
- [89] D. Piorkowski, M. Hind, and J. Richards, "Quantitative AI risk assessments: Opportunities and challenges," 2022, *arXiv:2209.06317*.
- [90] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," 2019, *arXiv:1910.09700*.
- [91] L. Lanelongue, J. Grealey, and M. Inouye, "Green algorithms: Quantifying the carbon footprint of computation," *Adv. Sci.*, vol. 8, no. 12, Jun. 2021, Art. no. 2100707.

- [92] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 13693–13696.
- [93] O. Shaikh, J. Saad-Falcon, A. P. Wright, N. Das, S. Freitas, O. I. Asensio, and D. H. Chau, "EnergyVis: Interactively tracking and exploring energy consumption for ML models," in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, 2021, pp. 1–7.
- [94] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, and J. Clause, "An empirical study of Practitioners' perspectives on green software engineering," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 237–248.
- [95] S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, and Y. Zou, "Green AI: Do deep learning frameworks have different costs?" in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng. (ICSE)*, May 2022, pp. 1082–1094.
- [96] R. Verdecchia, L. Cruz, J. Sallou, M. Lin, J. Wickenden, and E. Hotellier, "Data-centric green AI an exploratory empirical study," in *Proc. Int. Conf. ICT Sustainability (ICT4S)*, Plovdiv, Bulgaria, Jun. 2022, pp. 35–45.
- [97] A. Pachot and C. Patissier, "Towards sustainable artificial intelligence: An overview of environmental protection uses and issues," 2022, *arXiv:2212.11738*.
- [98] R. Vinuesa, H. Azizpour, I. Leite, M. Balaam, V. Dignum, S. Domisch, A. Felländer, S. D. Langhans, M. Tegmark, and F. Fuso Nerini, "The role of artificial intelligence in achieving the sustainable development goals," *Nature Commun.*, vol. 11, no. 1, p. 233, Jan. 2020.
- [99] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
- [100] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," 2019, *arXiv:1906.02243*.
- [101] J. Lin, X. Li, and G. Pekhimenko, "Multi-node BERT-pretraining: Cost-efficient approach," 2020, *arXiv:2008.00177*.
- [102] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune BERT for text classification?" 2019, *arXiv:1905.05583*.
- [103] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 64–77, Dec. 2020.
- [104] J. Lee, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Feb. 2020.
- [105] N. Poerner, U. Waltinger, and H. Schütze, "Inexpensive domain adaptation of pretrained language models: Case studies on biomedical NER and COVID-19 QA," 2020, *arXiv:2004.03354*.
- [106] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "AIBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.
- [107] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
- [108] B. Li, Z. Kong, T. Zhang, J. Li, Z. Li, H. Liu, and C. Ding, "Efficient transformer-based large scale language representations using hardware-friendly block structured pruning," 2020, *arXiv:2009.08065*.
- [109] A. Radford et al., *Improving Language Understanding by Generative Pre-Training*. San Francisco, CA, USA: OpenAI, 2018.
- [110] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [111] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," 2018, *arXiv:1809.11096*.
- [112] A. Chowdhery, "PaLM: Scaling language modeling with pathways," 2022, *arXiv:2204.02311*.
- [113] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "Fast deep neural network training on distributed systems and cloud TPUs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 11, pp. 2449–2462, Nov. 2019.
- [114] S. Chatterjee, A. M. Javid, M. Sadeghi, P. P. Mitra, and M. Skoglund, "Progressive learning for systematic design of large neural networks," 2017, *arXiv:1710.08177*.
- [115] M. Iman, H. R. Arabnia, and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technologies*, vol. 11, no. 2, p. 40, Mar. 2023.
- [116] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," 2020, *arXiv:2002.06305*.
- [117] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.
- [118] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 10882–11005, 2021.
- [119] Z. Ma, S. Edunov, and M. Auli, "A comparison of approaches to document-level machine translation," 2021, *arXiv:2101.11040*.
- [120] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala, "PyTorch distributed: Experiences on accelerating data parallel training," 2020, *arXiv:2006.15704*.
- [121] K. S. Chahal, M. S. Grover, K. Dey, and R. R. Shah, "A Hitchhiker's guide on distributed training of deep neural networks," *J. Parallel Distrib. Comput.*, vol. 137, pp. 65–76, Mar. 2020.
- [122] M. Rakka, M. E. Fouda, P. Khargonekar, and F. Kurdahi, "Mixed-precision neural networks: A survey," 2022, *arXiv:2208.06064*.
- [123] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, H. Peng, J. Li, J. Wu, Z. Liu, P. Xie, C. Xiong, J. Pei, P. S. Yu, and L. Sun, "A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT," 2023, *arXiv:2302.09419*.



ENRICO BARBIERATO received the B.S., M.Sc., and Ph.D. degrees in computer science. Currently, he is an Assistant Professor in data science with the Catholic University of the Sacred Heart, Brescia, Italy. He worked for 25 years in IT consulting for the banking, telecommunications, and energy and utilities industries. His research interests include performance evaluation through multi-formalism and ethical AI.



ALICE GATTI received the B.S. degree in mathematics and the M.S. degree in applied data science for banking and finance from the Catholic University of the Sacred Heart, Brescia, Italy.

Her research interests include artificial intelligence, green AI, and data fairness.

• • •