Sci

MDPI

Review

From Turing to Transformers: A Comprehensive Review and Tutorial on the Evolution and Applications of Generative Transformer Models

Emma Yann Zhang [1,*], Adrian David Cheok [2,*], Zhigeng Pan [1], Jun Cai [2] and Ying Yan [2]

[1] School of Artificial Intelligence, Nanjing University of Information Science and Technology, Nanjing 210044, China; zgpan@nuist.edu.cn
[2] School of Automation, Nanjing University of Information Science and Technology, Nanjing 210044, China; j.cai@nuist.edu.cn (J.C.); ying.yan@nuist.edu.cn (Y.Y.)
* Correspondence: 202351620003@nuist.edu.cn (E.Y.Z.); adrian@imagineeringinstitute.org (A.D.C.)

Abstract: In recent years, generative transformers have become increasingly prevalent in the field of artificial intelligence, especially within the scope of natural language processing. This paper provides a comprehensive overview of these models, beginning with the foundational theories introduced by Alan Turing and extending to contemporary generative transformer architectures. The manuscript serves as a review, historical account, and tutorial, aiming to offer a thorough understanding of the models' importance, underlying principles, and wide-ranging applications. The tutorial section includes a practical guide for constructing a basic generative transformer model. Additionally, the paper addresses the challenges, ethical implications, and future directions in the study of generative models.

Keywords: generative transformers; large language models; generative models; Alan Turing; artificial intelligence; machine learning; neural network; natural language processing

check for updates

Citation: Zhang, E.Y.; Cheok, A.D.; Pan, Z.; Cai, J.; Yan, Y. From Turing to Transformers: A Comprehensive Review and Tutorial on the Evolution and Applications of Generative Transformer Models. *Sci* **2023**, *5*, 46. https://doi.org/10.3390/sci5040046

Academic Editors: Carlo Cattani, Dioneia Motta Monte-Serrat, Francesco M. Donini and Paolo Bellavista

Received: 1 November 2023
Revised: 3 December 2023
Accepted: 11 December 2023
Published: 15 December 2023

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

1. Introduction

1.1. Background and Significance of Generative Models in AI

Generative models serve as an essential building block in the realm of artificial intelligence (AI). At their core, these models are designed to generate new data samples that are similar to the input data they have been trained on. This capability has profound implications, enabling machines to create, imagine, and replicate complex patterns observed in the real world.

The inception of generative models can be traced back to the early days of AI, where the foundational work of Alan Turing laid the groundwork for the evolution of generative models and the broader field of AI. Following Turing's pioneering contributions, the field witnessed the emergence of simple algorithms designed to mimic and reproduce sequential data. An exemplar of this era is the Hidden Markov Models (HMM) proposed by Leonard Baum in a series of seminal papers published in the late 1960s [1–3]. These models were groundbreaking for their time, providing a probabilistic framework to understand and predict sequences. The most notable application of HMMs was in the realm of speech recognition [4], where they became a foundational component, enabling systems to decode and understand human speech with increasing accuracy.

The introduction of Recurrent Neural Networks (RNNs) in 1982 by John Hopfield [5] and Long Short-Term Memory (LSTM) networks in 1997 by Hochreiter and Schmidhuber [6] marked significant advancements in the field. RNNs brought the ability to remember previous inputs in handling sequential data, while LSTMs addressed the challenges of long-term dependencies, making them pivotal for tasks such as time series prediction, speech recognition, and natural language processing. Together, they set foundational standards for modern generative AI models handling sequences.

FriendliAI Inc. / Hugging Face, Inc.
Exhibit 2008
Page 1

However, with the advent of deep learning and the proliferation of neural networks, the potential and capabilities of generative models have expanded exponentially. Neural-based generative models, such as Variational Autoencoders (VAEs) [7,8] introduced in 2013 and Generative Adversarial Networks (GANs) [9,10] introduced in the following year, have showcased the ability to generate high-fidelity new data samples based on training data, ranging from images to text and even music.

The significance of generative models in AI is multifaceted. Firstly, they play a pivotal role in unsupervised learning, where labeled data is scarce or unavailable. By learning the underlying distribution of the data, generative models can produce new samples, aiding in tasks such as data augmentation [11,12], anomaly detection [13], and image denoising [14,15]. Secondly, the creative potential of these models has been harnessed in various domains, from image [16–19], video, and music generation to drug discovery [20,21] and virtual reality [22–24]. The ability of machines to generate novel and coherent content has opened up avenues previously deemed exclusive to human creativity.

Furthermore, generative models serve as powerful tools for understanding and interpreting complex data distributions. They provide insights into the structure and relationships within the data, enabling researchers and practitioners to uncover hidden patterns, correlations, and features [25]. This interpretative power is especially valuable in domains such as biology [26], finance [27], and climate science [28], where understanding data intricacies can lead to groundbreaking discoveries.

Generative models stand as a testament to the advancements and possibilities within AI. Their ability to create, interpret, and innovate has not only broadened the horizons of machine learning but has also reshaped our understanding of intelligence and creativity.

1.2. The Rise of Transformer Architectures

While Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) have significantly advanced the field of generative AI, another monumental shift in the deep learning landscape emerged with the introduction of the transformer architecture. Presented in the seminal paper "Attention is All You Need" by a team of Google researchers led by Vaswani in 2017 [29], transformers have redefined the benchmarks in a multitude of tasks, particularly in natural language processing (NLP).

The transformer's innovation lies in its self-attention mechanism, which allows it to weigh the significance of different parts of an input sequence, be it words in a sentence or pixels in an image. This mechanism enables the model to capture long-range dependencies and intricate relationships in the data, overcoming the limitations of previous architectures such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. RNNs and LSTMs, while effective in handling sequential data, often struggled with long sequences due to issues such as vanishing and exploding gradients [30]. Transformers, with their parallel processing capabilities and attention mechanisms, alleviated these challenges.

The success of the transformer architecture was not immediate but became evident with the introduction of large language models such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer). BERT, developed by researchers at Google, demonstrated the power of transformers in understanding the context of words in a sentence by considering both left and right contexts in all layers [31]. This bidirectional approach led to state-of-the-art results in several NLP tasks, from question answering to sentiment analysis [32]. On the other hand, OpenAI's GPT showcased the generative capabilities of transformers [33], producing human-like text and achieving remarkable performance in tasks such as machine translation [34] and text summarization [35] without task-specific training data.

The transformer's versatility extends beyond NLP. Vision Transformer (ViT) [36], an adaptation of the architecture for image classification tasks, has shown that transformers can rival, if not surpass, the performance of traditional convolutional neural networks

(CNNs) in computer vision tasks [37,38]. This cross-domain applicability underscores the transformer's potential and its foundational role in modern AI.

Another driving factor behind the rise of transformers is the ever-growing computational power and the availability of large-scale datasets. Training transformer models, especially large ones, require significant computational resources. The feasibility of training such models has been made possible due to advancements in GPU and TPU technologies [39], coupled with the availability of vast amounts of data to train on. The combination of innovative architecture and computational prowess has led to the development of models with billions or even trillions of parameters, pushing the boundaries of what machines can generate to new heights.

Generative AI models have undergone significant transformations since their inception, with each milestone contributing to the capabilities we see today. From the foundational Turing machines to the latest GPT-4 and LLaMA models, the journey of generative AI has been marked by groundbreaking advancements. A detailed timeline capturing these key milestones is presented to offer a comprehensive overview of the field's evolution (Figure 1).

1.3. Purpose and Structure of the Paper

The fast growth in artificial intelligence, especially with recent technologies such as generative models and transformers, highlights the need for a comprehensive study that spans both their historical development and current applications. The primary objective of this paper is to provide readers with a holistic understanding of the evolution, significance, architecture, and capabilities of generative transformers, contextualized within the broader landscape of AI.

Our motivation for this paper is informed by the existing body of work on transformer-based models and generative AI. While there are several comprehensive reviews, each focuses on specific aspects of the topic. For example, Gozalo-Brizuela and Garrido-Merchan [40] concentrate on the taxonomy and industrial implications of large generative models, providing a compilation of popular generative models organized into various categories such as text-to-text, text-to-image, and text-to-audio. Lin et al. [41] present an exhaustive review of various transformer variants, their architectural modifications, and applications. Additionally, there are survey papers that focus on the use of transformers for specific tasks such as natural language processing [42,43], computer vision [44–47], time series analysis and forecasting [48,49], among others. These existing reviews are invaluable, but our paper aims to provide a more comprehensive overview that bridges these specialized areas.

While these papers offer valuable insights, there is a gap in the literature for a resource that combines a historical review, a hands-on tutorial, and a forward-looking perspective on generative transformer models. Our paper aims to fill this void, serving as a comprehensive guide for newcomers and seasoned researchers alike. The historical review section helps readers understand how generative AI has developed and progressed in the wider context of AI. Meanwhile, our practical tutorial guides readers through the foundational concepts and practical implementations, equipping them to build their own generative transformer models. We offer a unique blend of theoretical understanding and practical know-how, setting our work apart from existing reviews. Additionally, we strive to provide a unique balance between explaining the historical evolution, technical aspects, and applications of transformers. This makes our paper a go-to source for researchers and professionals seeking a wholesome understanding and knowledge of transformers.

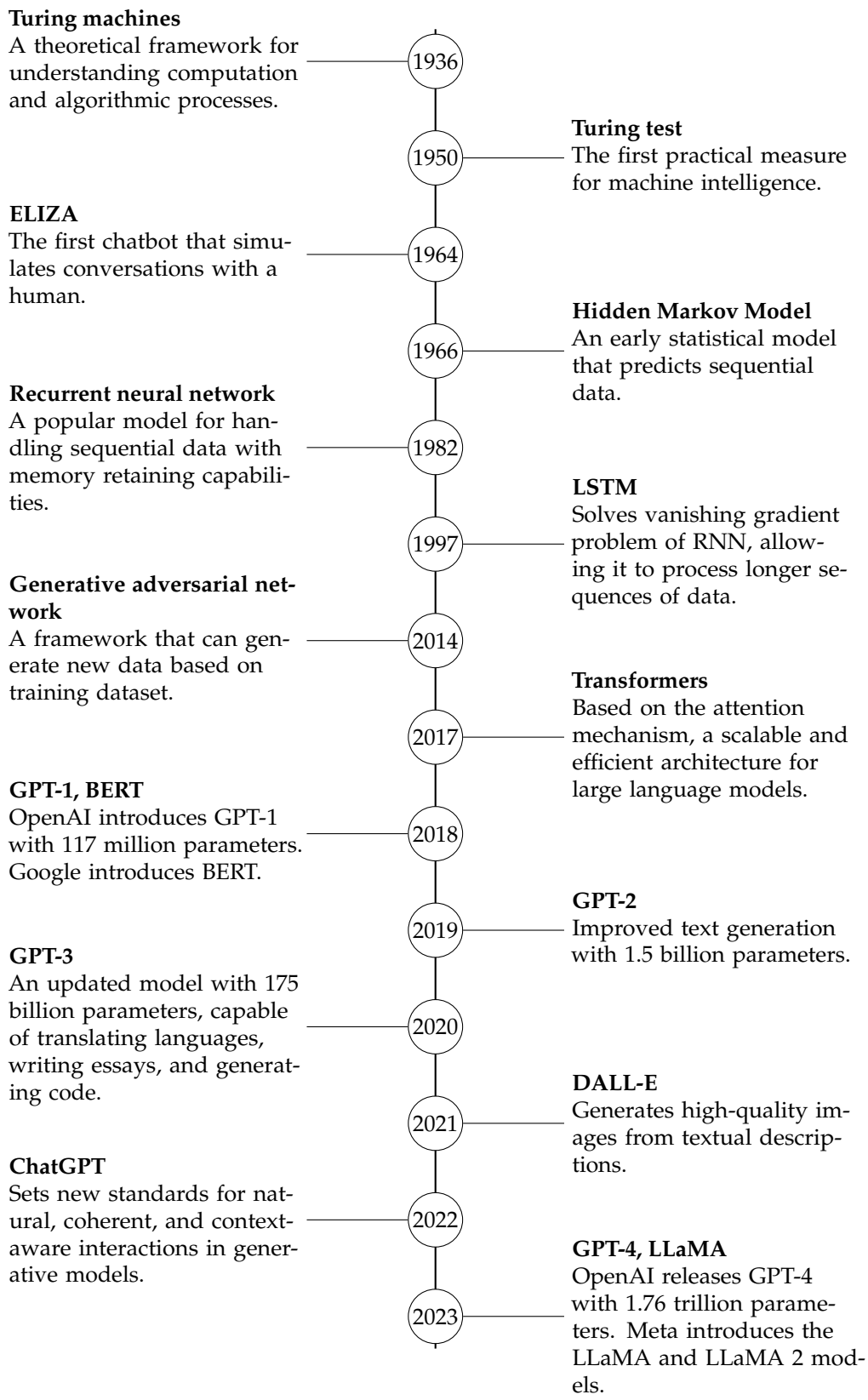

Figure 1. A timeline illustrating key milestones in the development of generative AI, from Turing Machines to GPT-4.

The structure of the paper, which is designed to guide the reader through a logical progression, is as follows:

FriendliAI Inc. / Hugging Face, Inc.
Exhibit 2008
Page 4

- Historical Evolution: We embark on a journey tracing the roots of computational theory, starting with the foundational concepts introduced by Alan Turing. This section provides a backdrop, setting the stage for the emergence of neural networks, the challenges they faced, and the eventual rise of transformer architectures.
- Tutorial on Generative Transformers: Transitioning from theory to practice, this section offers a practical approach to understanding the intricacies of generative transformers. Readers will gain insights into the architecture, training methodologies, and best practices, supplemented with code snippets and practical examples.
- Applications and Challenges: Building upon the foundational knowledge, we delve into the myriad applications of generative transformers, highlighting their impact across various domains. Concurrently, we address the challenges and ethical considerations associated with their use, fostering a balanced perspective.
- Conclusion and Future Directions: The paper concludes with a reflection on the current state of generative transformers, their potential trajectory, and the exciting possibilities they hold for the future of AI.

In essence, this paper endeavors to be more than just a review or a tutorial, it aspires to be a comprehensive guide, weaving together history, theory, practice, and prospects, providing readers with a panoramic view of the world of generative transformers.

2. Historical Evolution

The development of computational theory and artificial intelligence has been shaped by pioneering figures, innovative ideas, and transformative discoveries. Central to this narrative is Alan Turing, whose unparalleled contributions laid the foundations for modern computation and the subsequent emergence of AI. This section delves deeper into Turing's groundbreaking work, and the lasting legacy that continues to shape the digital age.

2.1. Turing Machines and the Foundations of Computation

One of Turing's major contributions was the idea of the Turing machine proposed in his 1936 paper titled "On Computable Numbers, with an Application to the Entscheidungsproblem" [50]. This abstract machine was a simple but powerful theoretical construct that was designed to perform computations by manipulating symbols on an infinite tape based on a set of rules. The infinite tape is divided into discrete cells, each cell can contain a symbol from a finite alphabet, and the machine itself has a "head" that can read and write symbols on the tape and move left or right. The machine's behavior is dictated by a set of transition rules, which determine its actions based on the current state and the symbol being read. In essence, the Turing machine is a rule-based system that manipulates symbols on a tape, embodying the fundamental operations of reading, writing, and transitioning between states.

While the concept might seem rudimentary, the implications of the Turing machine are profound. Turing demonstrated that this simple device, with its set of rules and operations, could compute any function that is computable, given enough time and tape. This assertion, known as the Church–Turing thesis [51] (independently proposed by Alonzo Church in his paper titled "An Unsolvable Problem of Elementary Number Theory" also published in 1936 [52]), posits that any function computable by an algorithm can be computed by a Turing machine. This thesis, although not proven, has stood the test of time, with no evidence to the contrary. It serves as a foundational pillar in computer science, defining the boundaries of what is computable.

World War II saw Turing's theoretical concept manifest in tangible, real-world applications. Stationed at Bletchley Park, Britain's cryptographic hub, Turing played a key role in deciphering the Enigma code used by the German military. Turing helped develop a machine called the Bombe, which expedited the decryption process of Enigma-encrypted messages [53]. This secret work was crucial for the Allies' success and showed how computer science could have a major impact on real-world events.

After World War II, Turing turned his attention to the development of electronic computers. He was instrumental in the design of the Automatic Computing Engine (ACE) [54], one of the earliest computer models capable of storing programs. This showed Turing's forward-thinking approach to the digital age. Beyond computing, he also delved into the nature of intelligence and how it could be replicated in machines.

The Turing machine's significance transcended its immediate mathematical implications. The true brilliance of Turing's insight, however, lies in the concept of universal computation. Turing's subsequent proposition of a Universal Turing Machine (UTM)—a machine capable of simulating any other Turing machine given the right input and rules—was a revolutionary idea [50]. Given a description of a Turing machine and its input encoded on the tape, the UTM could replicate the behavior of that machine. This meta-level of computation was groundbreaking. It suggested that a single, general-purpose machine could be designed to perform any computational task, eliminating the need for task-specific machines. The UTM was a harbinger of modern computers, devices that can be reprogrammed to execute a wide array of tasks.

The implications of universal computation extend beyond mere hardware. It challenges our understanding of intelligence and consciousness. If the human brain, with its intricate neural networks and synaptic connections, operates on computational principles, then could it be simulated by a Turing machine? This question, which blurs the lines between philosophy, neuroscience, and computer science, remains one of the most intriguing and debated topics in the field of artificial intelligence.

2.1.1. Turing's Impact on Artificial Intelligence and Machine Learning

Alan Turing's influence on the fields of artificial intelligence (AI) and machine learning (ML) is both profound and pervasive. While Turing is often lauded for his foundational contributions to computational theory, his vision and insights into the realm of machine intelligence have played a pivotal role in shaping the trajectory of AI and ML.

His 1950 paper, "Computing Machinery and Intelligence", Ref. [55] introduced the famous Turing Test as a practical measure of machine intelligence. Alan Turing introduced the Turing Test within the context of an "Imitation Game", involving a man, a woman, and a judge as players. They communicate electronically from separate rooms, and the goal of the judge is to identify who is the woman. The man aims to deceive the judge into thinking he is the woman, while the woman assists the judge. Turing then adapts this game into his famous test by replacing the man with a machine, aiming to deceive the questioner in the same way. Although the original game focused on gender identification, this aspect is often overlooked in later discussions of the Turing Test.

In this work, Turing posed the provocative question: "Can machines think?" Rather than delving into the philosophical intricacies of defining "thinking", Turing proposed a pragmatic criterion for machine intelligence: if a machine could engage in a conversation with a human, indistinguishably from another human, it would be deemed intelligent. This criterion, while straightforward, sparked widespread debate and research, laying the foundation for the field of artificial intelligence.

The Turing Test, in many ways, encapsulated the essence of AI—the quest to create machines that can mimic, replicate, or even surpass human cognitive abilities. It set a benchmark, a gold standard for machine intelligence, challenging researchers and scientists to build systems that could "think" and "reason" like humans. While the test itself has been critiqued and refined over the years, its underlying philosophy remains central to AI: the aspiration to understand and emulate human intelligence.

Beyond the Turing Test, Turing's insights into neural networks and the potential of machine learning were visionary. In a lesser-known report written in 1948, titled "Intelligent Machinery" [56], Turing delved into the idea of machines learning from experience. He envisioned a scenario where machines could be trained, much like a human child, through a process of education. Turing postulated the use of what he termed "B-type unorganized machines", which bear a striking resemblance to modern neural networks. These machines,

as Turing described, would be trained, rather than explicitly programmed, to perform tasks. Although in its infancy at the time, this idea signaled the rise of machine learning, where algorithms learn from data rather than being explicitly programmed.

Turing's exploration of morphogenesis, the biological process that causes organisms to develop their shape, further showcased his interdisciplinary genius [57]. In his work on reaction-diffusion systems, Turing demonstrated how simple mathematical models could give rise to complex patterns observed in nature. This work, while primarily biological in its focus, has profound implications for AI and ML. It underscores the potential of simple algorithms to generate complex, emergent behavior, a principle central to neural networks and deep learning.

Alan Turing's impact on artificial intelligence and machine learning is immeasurable. His vision of machine intelligence, his pioneering insights into neural networks, and his interdisciplinary approach to problem-solving have left an indelible mark on the field. As we navigate the intricate landscape of modern AI, with its deep neural networks, generative models, and transformers, it is imperative to recognize and honor Turing's legacy. His work serves as a beacon, illuminating the path forward, reminding us of the possibilities, challenges, and the profound potential of machines that can "think".

2.1.2. From Turing's Foundations to Generative Transformers

The journey from Alan Turing's foundational concepts to the sophisticated realm of generative transformers is a testament to the evolution of computational theory and its application in artificial intelligence. While at first glance Turing's work and generative transformers might seem worlds apart, a closer examination reveals a direct lineage and influence.

Alan Turing's conceptualization of the Turing machine provided the bedrock for understanding computation. His idea of a machine that could simulate any algorithm, given the right set of instructions, laid the groundwork for the concept of universal computation. This idea, that a single machine could be reprogrammed to perform a myriad of tasks, is the precursor to the modern notion of general-purpose computing systems.

Fast forward to the advent of neural networks, which Turing had touched upon in his lesser-known works. These networks, inspired by the human brain's interconnected neurons, were designed to learn from data. The foundational idea was that, rather than being explicitly programmed to perform a task, these networks would "learn" by adjusting their internal parameters based on the data they were exposed to. Turing's vision of machines learning from experience resonates deeply with the principles of neural networks.

Generative transformers, a cutting-edge development in the AI landscape, are an extension of these neural networks. Transformers, with their self-attention mechanisms, are designed to weigh the significance of different parts of an input sequence, capturing intricate relationships within the data. The "generative" aspect of these models allows them to produce new, previously unseen data samples based on their training.

Drawing a direct link, Turing's Universal Turing Machine can be seen as an early, abstract representation of what generative transformers aim to achieve in a more specialized domain. Just as the Universal Turing Machine could simulate any other Turing machine, given the right input and set of rules, generative transformers aim to generate any plausible data sample, given the right training and context. The universality of Turing's machine finds its parallel in the versatility of generative transformers.

Furthermore, Turing's exploration into machine learning, the idea of machines learning from data rather than explicit programming, is the very essence of generative transformers. These models are trained on vast datasets, learning patterns, structures, and nuances, which they then use to generate new content. The bridge between Turing's early insights into machine learning and the capabilities of generative transformers is a direct one, showcasing the evolution of a concept from its theoretical inception to its practical application.

While Alan Turing might not have directly worked on generative transformers, his foundational concepts, vision of machine learning, and the principles he laid down have

directly influenced and shaped their development. The journey from Turing machines to generative transformers is a testament to the enduring legacy of Turing's genius and the continual evolution of artificial intelligence.

2.2. Early Neural Networks and Language Models

The realm of artificial intelligence has witnessed a plethora of innovations and advancements, with neural networks standing at the forefront of this revolution. These computational models, inspired by the intricate web of neurons in the human brain, have paved the way for sophisticated language models that can understand, generate, and manipulate human language with unprecedented accuracy.

2.2.1. Introduction to Neural Networks

Neural networks [58,59], at their core, are a set of algorithms designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling, and clustering of raw input. These algorithms loosely mirror the way a human brain operates, thus the nomenclature "neural networks".

A basic neural network consists of layers of interconnected nodes or "neurons". Each connection between neurons has an associated weight, which is adjusted during training. The fundamental equation governing the output y of a neuron is given by:

$$y = f\left(\sum_i w_i x_i + b\right) \tag{1}$$

where x_i are the input values, w_i are the weights, b is a bias term, and f is an activation function.

The activation function introduces non-linearity into the model, allowing it to learn from error and make adjustments, which is essential for learning complex patterns. One of the commonly used activation functions is the sigmoid function, defined as:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

Neural networks typically consist of an input layer, one or more hidden layers, and an output layer. The depth and complexity of a network, often referred to as its "architecture", determine its capacity to learn from data.

2.2.2. Evolution of Recurrent Neural Networks (RNNs)

While traditional neural networks have proven effective for a wide range of tasks, they possess inherent limitations when dealing with sequential data. This is where Recurrent Neural Networks (RNNs) come into play. RNNs are designed to recognize patterns in sequences of data, such as time series or natural language.

The fundamental difference between RNNs and traditional neural networks lies in the former's ability to retain memory of previous inputs in its internal state. This is achieved by introducing loops in the network, allowing information to persist.

The output of an RNN at time t, denoted h_t, is computed as:

$$h_t = f(W_{hh} h_{t-1} + W_{xh} x_t + b) \tag{3}$$

where W_{hh} and W_{xh} are weight matrices, x_t is the input at time t, and h_{t-1} is the output from the previous timestep.

While RNNs are powerful, they suffer from challenges such as the vanishing and exploding gradient problems, especially when dealing with long sequences [30]. This makes them less effective in capturing long-term dependencies in the data.

2.2.3. Long Short-Term Memory (LSTM) Networks

To address the vanishing gradient problem of RNNs, Long Short-Term Memory (LSTM) networks were introduced. LSTMs, a special kind of RNN, are designed to remember information for extended periods [60].

The core idea behind LSTMs is the cell state, a horizontal line running through the entire chain of repeating modules in the LSTM. The cell state can carry information from earlier time steps to later ones, mitigating the memory issues faced by traditional RNNs.

LSTMs introduce three gates:

1. Forget Gate: It decides what information from the cell state should be thrown away or kept. Mathematically, the forget gate f_t is given by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4}$$

2. Input Gate: It updates the cell state with new information. The input gate i_t and the candidate values $\tilde{C}_t$ are computed as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{6}$$

3. Output Gate: It determines the output based on the cell state and the input. The output h_t is given by:

$$h_t = o_t \times \tanh(C_t) \tag{7}$$

where o_t is the output gate, defined as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{8}$$

LSTMs, with their ability to capture long-term dependencies and mitigate the challenges faced by traditional RNNs, have paved the way for advancements in sequence modeling, particularly in the domain of natural language processing.

2.3. The Advent of Transformers

In the ever-evolving landscape of artificial intelligence and machine learning, the transformer architecture stands out as a significant leap forward, especially in the domain of natural language processing. Introduced in the seminal paper "Attention Is All You Need" by Vaswani et al. [29], transformers have revolutionized the way we approach sequence-to-sequence tasks. This section aims to demystify the transformer architecture, breaking it down into its core components and principles.

2.3.1. Introduction to the Transformer Architecture

At a high level, the transformer is a type of neural network architecture designed to handle sequential data, making it particularly well-suited for tasks such as language translation, text generation, and more. Unlike its predecessors, such as RNNs and LSTMs, which process data in order, transformers leverage a mechanism called "attention" to draw global dependencies between input and output.

The heart of the transformer architecture is the attention mechanism. In essence, attention allows the model to focus on different parts of the input sequence when producing an output sequence, much like how humans pay attention to specific words when understanding a sentence.

Mathematically, the attention score for a given query q and key k is computed as:

$$\text{Attention}(q, k) = \frac{\exp(\text{score}(q, k))}{\sum_{k'} \exp(\text{score}(q, k'))} \tag{9}$$

where score is a function that calculates the relevance of the key k to the query q. The output of the attention mechanism is a weighted sum of values, where the weights are the attention scores.

The transformer model consists of an encoder and a decoder. Each of these is composed of multiple layers of attention and feed-forward neural networks.

The encoder takes in a sequence of embeddings (representations of input tokens) and processes them through its layers. The decoder then generates the output sequence, leveraging both its internal layers and the encoder's output.

One of the distinguishing features of transformers is the use of "multi-head attention", which allows the model to focus on different parts of the input simultaneously, capturing various aspects of the information.

2.3.2. Advantages of Transformers

Transformers have brought significant advancements in the processing of sequential data, characterized by several key advantages. One notable feature of transformers is parallelization. Unlike RNNs, which process sequences step-by-step, transformers can process all tokens in parallel, leading to faster training times.

Transformers are also known for their adeptness at handling long-range dependencies. The attention mechanism enables transformers to capture relationships between tokens, regardless of their distance in the sequence. This capability is particularly beneficial for complex tasks where context and relationships between distant elements are crucial for accurate interpretation and response.

Scalability is another advantage of transformer models. Transformers are highly scalable, making them well-suited for dealing with large datasets and intricate tasks. This scalability ensures that transformers remain effective and efficient even as the size and complexity of the data or the task increase.

2.4. Attention Mechanism: The Heart of Transformers

The attention mechanism, a pivotal innovation in the realm of deep learning, has transformed the way we approach sequence-to-sequence tasks in natural language processing. Serving as the cornerstone of the transformer architecture, attention allows models to dynamically focus on different parts of the input data, capturing intricate relationships and dependencies. This section aims to elucidate the principles and mathematics behind the attention mechanism, shedding light on its significance in the transformer architecture.

2.4.1. Conceptual Overview of Attention

In traditional sequence-to-sequence models, such as RNNs and LSTMs, information from the entire input sequence is compressed into a fixed-size context vector, which is then used to generate the output sequence. This approach, while effective for short sequences, struggles with longer sequences as the context vector becomes a bottleneck, unable to capture all the nuances of the input data.

The attention mechanism addresses this challenge by allowing the model to "attend" to different parts of the input sequence dynamically, based on the current context. Instead of relying on a single context vector, the model computes a weighted sum of all input vectors, where the weights represent the "attention scores".

2.4.2. Mathematics of Attention

The core of the attention mechanism is the computation of attention scores. Given a query q and a set of key-value pairs (k, v), the attention score for a specific key k is computed as:

$$\text{score}(q, k) = q^T k \tag{10}$$

The attention weights, which determine how much focus should be given to each key-value pair, are computed using a softmax function:

$$\text{Attention}(q, k) = \frac{\exp(\text{score}(q, k))}{\sum_{k'} \exp(\text{score}(q, k'))} \tag{11}$$

The output of the attention mechanism is a weighted sum of the values:

$$\text{output} = \sum_i \text{Attention}(q, k_i) v_i \tag{12}$$

As depicted in Figure 2, the attention mechanism computes scores based on the query and keys, derives attention weights, and produces an output based on a weighted sum of values.

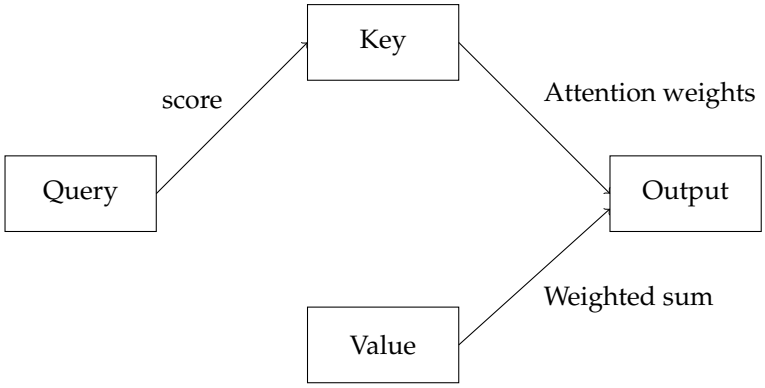

Figure 2. Schematic representation of the attention mechanism.

2.4.3. Significance in Transformers

In the transformer architecture, attention is not just a supplementary feature; it is the core component. Transformers employ a variant called "multi-head attention", which runs multiple attention mechanisms in parallel, capturing different types of relationships in the data.

The attention mechanism's ability to focus on different parts of the input sequence, irrespective of their position, empowers transformers to handle long-range dependencies, making them particularly effective for tasks like language translation, text summarization, and more.

Furthermore, the self-attention mechanism, a special case where the query, key, and value are all derived from the same input, enables transformers to weigh the significance of different parts of the input relative to a specific position. This is crucial for understanding context and semantics in natural language processing tasks.

2.5. Generative Transformers and Their Significance

Generative transformers have emerged as a groundbreaking advancement in the domain of artificial intelligence, particularly in natural language processing and generation. These models, characterized by their ability to generate coherent and contextually relevant sequences of text, have set new benchmarks in various tasks, from text completion to story generation. This section introduces the notable generative models available, including the GPT series and other significant contributions in this domain.

2.5.1. GPT (Generative Pre-Trained Transformer) Series

The GPT series, developed by OpenAI, fully demonstrates the power and potential of generative transformers. Built upon the transformer architecture, the GPT models leverage the attention mechanism to understand and generate human-like text. The GPT series has seen rapid evolution, with each iteration bringing enhanced capabilities and performance.

GPT-1. The first in the series, GPT-1 [61], was released in 2018. It laid the foundation for subsequent models. With 117 million parameters, it showcased the potential of transformers in generating coherent paragraphs of text.

GPT-2. Released in 2019, GPT-2 [62] increased its parameters to 1.5 billion. Its ability to generate entire articles, answer questions, and even write poetry garnered significant attention from the research community and the public alike.

GPT-3. GPT-3 [63] has 175 billion parameters. Its capabilities extend beyond mere text generation; it can translate languages, write essays, create poetry, and even generate code.

GPT-4. The most recent model from OpenAI, GPT-4 [64], consists a staggering 1.76 trillion parameters, positioning it among the most advanced language models currently available. Leveraging advanced deep learning methodologies, it surpasses the capabilities of its forerunner, GPT-3. Remarkably, GPT-4 can handle up to 25,000 words simultaneously, a capacity 8-fold greater than GPT-3. Furthermore, GPT-4 is versatile in accepting both text and image prompts, allowing users to define tasks across vision and language domains. A notable improvement in GPT-4 is its reduced propensity for hallucinations compared to earlier versions.

2.5.2. Other Notable Generative Transformer Models

Beyond the GPT series, the landscape of generative transformers is rich and diverse, with several models making significant contributions to the field.

BERT (Bidirectional Encoder Representations from Transformers). Developed by Google, BERT [31] revolutionized the way we approach natural language understanding tasks. Unlike GPT, which is generative, BERT is discriminative, designed to predict missing words in a sentence. Its bidirectional nature allows it to capture context from both the left and the right of a word, leading to superior performance in tasks like question-answering and sentiment analysis.

LLaMA. LLaMA [65] is an auto-regressive language model built on the transformer architecture, introduced by Meta. In February 2023, Meta unveiled the initial version of LLaMA, boasting 65 billion parameters and adept at numerous generative AI functions. By July 2023, LLaMA 2 was launched with 3 distinct model sizes: 7, 13, and 70 billion parameters.

LaMDA. LaMDA [66] is a specialized family of transformer-based neural language models for dialog applications developed by Google in 2022. With up to 137 billion parameters and pre-training on 1.56 trillion words of public dialog and web text, LaMDA aims to address two key challenges: safety and factual grounding. The model incorporates fine-tuning and external knowledge consultation to improve its safety metrics, ensuring responses align with human values and avoid harmful or biased suggestions. For factual grounding, LaMDA employs external knowledge sources like information retrieval systems and calculators to generate responses that are not just plausible but also factually accurate. The model shows promise in various domains, including education and content recommendations, offering a balanced blend of quality, safety, and factual integrity.

3. Tutorial on Generative Transformers

In this section, we delve into a hands-on tutorial on generative transformers, guiding readers through the foundational concepts and practical implementations. By the end of this tutorial, readers should have a clear understanding of the transformer architecture and be equipped to build their own generative transformer models.

3.1. Basics of the Transformer Architecture

The transformer architecture, introduced by Vaswani et al. in their seminal paper "Attention Is All You Need" [29], has become the backbone of many state-of-the-art models in natural language processing. We will now break down its core components.

3.1.1. Overview

As depicted in Figure 3, the transformer consists of an encoder and a decoder. The encoder processes the input sequence, and the decoder generates the output sequence.

Both the encoder and decoder are composed of multiple layers of attention mechanisms and feed-forward neural networks.

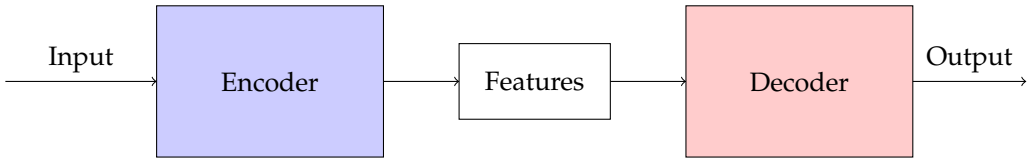

Figure 3. Expanded schematic representation of the transformer architecture with a smaller Features block.

3.1.2. Attention Mechanism

As previously discussed, the attention mechanism allows the model to focus on different parts of the input sequence when producing an output. The mechanism computes attention scores based on queries, keys, and values.

Mathematical Representation:

Given a query q, key k, and value v, the attention output is computed as:

$$\text{Attention}(q, k, v) = \text{softmax}\left(\frac{q \cdot k^T}{\sqrt{d_k}}\right)v \tag{13}$$

where d_k is the dimension of the key.

Code Snippet: The following Python code snippet demonstrates how to implement this attention mechanism using PyTorch:

```
import torch
import torch.nn.functional as F

def scaled_dot_product_attention(q, k, v):
    matmul_qk = torch.matmul(q, k.transpose(-2, -1))
    d_k = q.size(-1) ** 0.5
    scaled_attention_logits = matmul_qk / d_k
    attention_weights = F.softmax(scaled_attention_logits, dim=-1)
    output = torch.matmul(attention_weights, v)
    return output, attention_weights
```

In this code snippet, q, k, and v are the query, key, and value tensors, respectively. The function *scaled_dot_product_attention* computes the attention output according to Equation (13).

3.1.3. Multi-Head Attention

Instead of using a single set of attention weights, the transformer uses multiple sets, allowing it to focus on different parts of the input simultaneously. This is known as multi-head attention.

Code Snippet:

```
class MultiHeadAttention(nn.Module):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.num_heads = num_heads
        # Dimension of the model
        self.d_model = d_model
        # Depth of each attention head
        self.depth = d_model
        # Linear layer for creating query, key and value matrix
        self.wq = nn.Linear(d_model, d_model)
```

```
        self.wk = nn.Linear(d_model, d_model)
        self.wv = nn.Linear(d_model, d_model)
        # Final linear layer to produce the output
        self.dense = nn.Linear(d_model, d_model)
```

3.1.4. Feed-Forward Neural Networks

Each transformer layer contains a feed-forward neural network, applied independently to each position.

Code Snippet:

```
class PointWiseFeedForwardNetwork(nn.Module):
    def __init__(self, d_model, dff):
        super(PointWiseFeedForwardNetwork, self).__init__()
        self.fc1 = nn.Linear(d_model, dff)
        self.fc2 = nn.Linear(dff, d_model)
        ...
```

Each method and its body are indented with a tab or four spaces, which is the standard Python indentation. This makes the code easier to read and understand.

3.1.5. Self-Attention Mechanism

The self-attention mechanism is a variant of the attention mechanism where the input sequence itself serves as the queries, keys, and values. This allows the transformer to weigh the significance of different parts of the input relative to a specific position, crucial for understanding context and semantics.

Mathematical Representation:

Given an input sequence X, the queries Q, keys K, and values V are derived as:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{14}$$

where W_Q, W_K, and W_V are weight matrices. The self-attention output is then computed using the attention formula:

$$\text{SelfAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{15}$$

3.1.6. Positional Encoding

Transformers, by design, do not have a built-in notion of sequence order. To provide the model with positional information, we inject positional encodings to the input embeddings. These encodings are added to the embeddings to ensure the model can make use of the sequence's order.

Mathematical Representation:

The positional encodings are computed using sine and cosine functions:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{16}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{17}$$

where pos is the position and i is the dimension.

3.1.7. Multi-Head Attention

Multi-head attention is an extension of the attention mechanism, allowing the model to focus on different parts of the input simultaneously. By running multiple attention mechanisms in parallel, the model can capture various types of relationships in the data.

Mathematical Representation:

Given queries Q, keys K, and values V, the multi-head attention output is computed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W_O \tag{18}$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_{Qi}, KW_{Ki}, VW_{Vi}) \tag{19}$$

and W_{Qi}, W_{Ki}, W_{Vi}, and W_O are weight matrices.

Figure 4 showcases the multi-head attention mechanism, where multiple attention heads operate in parallel, and their outputs are concatenated and passed through a dense layer to produce the final output.

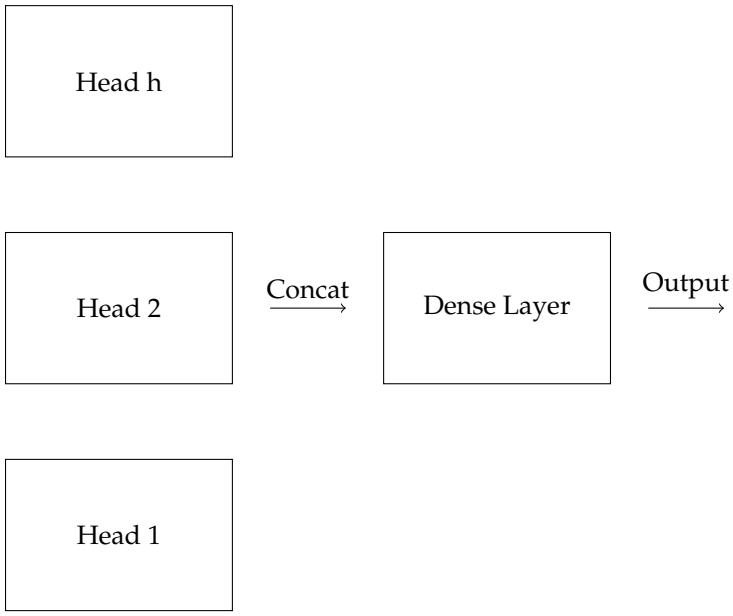

Figure 4. Schematic representation of multi-head attention.

Understanding the intricacies of the transformer architecture, from the self-attention mechanism to multi-head attention, is crucial for harnessing its full potential. By delving into the mathematical foundations and practical implementations, one can build powerful models capable of handling a wide range of tasks in natural language processing.

3.1.8. Encoder and Decoder Modules

The Transformer architecture consists of an encoder and a decoder, each made up of multiple layers. Here, we'll walk through the implementation of these modules.

Encoder Module. The encoder module consists of multiple encoder layers, each containing multi-head attention and feed-forward neural networks.

Code Snippet:

```
import torch.nn as nn

class EncoderLayer(nn.Module):
    def __init__(self, d_model, num_heads):
        super(EncoderLayer, self).__init__()
        self.mha = MultiHeadAttention(d_model, num_heads)
        self.ffn = PointWiseFeedForwardNetwork(d_model, dff)
        # Layer normalization and dropout layers can be added here

    def forward(self, x):
```

```
        attn_output = self.mha(x, x, x)
        out1 = x + attn_output  # Add & Norm
        ffn_output = self.ffn(out1)
        out2 = out1 + ffn_output  # Add & Norm
        return out2
```

Decoder Module. The decoder module is similar to the encoder but has an additional multi-head attention layer to attend to the encoder's output.

Code Snippet:

```
class DecoderLayer(nn.Module):
    def __init__(self, d_model, num_heads):
        super(DecoderLayer, self).__init__()
        self.mha1 = MultiHeadAttention(d_model, num_heads)
        self.mha2 = MultiHeadAttention(d_model, num_heads)
        self.ffn = PointWiseFeedForwardNetwork(d_model, dff)
        # Layer normalization and dropout layers can be added here

    def forward(self, x, enc_output):
        attn1 = self.mha1(x, x, x)
        out1 = x + attn1  # Add & Norm
        attn2 = self.mha2(out1, enc_output, enc_output)
        out2 = out1 + attn2  # Add & Norm
        ffn_output = self.ffn(out2)
        out3 = out2 + ffn_output  # Add & Norm
        return out3
```

In these code snippets, 'MultiHeadAttention' and 'PointWiseFeedForwardNetwork' are custom classes that you would define based on your specific needs for multi-head attention and point-wise feed-forward networks, respectively.

3.2. Building a Simple Generative Transformer

Building a generative transformer from scratch involves several steps, from data preprocessing to model training and text generation. In this section, we'll walk through each of these steps, providing a comprehensive guide to constructing your own generative transformer.

3.2.1. Data Preprocessing and Tokenization

Before feeding data into the model, it is essential to preprocess and tokenize it. Tokenization involves converting raw text into a sequence of tokens, which can be words, subwords, or characters.

Using popular libraries like the HuggingFace's 'transformers', tokenization can be achieved as:

```
from transformers import GPT2Tokenizer

tokenizer = GPT2Tokenizer.from_pretrained('gpt2-medium')
tokens = tokenizer.encode("Hello, world!")
```

3.2.2. Defining the Transformer Model

Assuming one has already defined the `EncoderLayer` and `DecoderLayer` classes, one can define the complete Transformer model as follows:

```
class Transformer(nn.Module):
        def __init__(self, d_model, num_heads, num_layers):
            super(Transformer, self).__init__()
            self.encoder = nn.ModuleList([EncoderLayer(d_model, num_heads) for _ in range(num_layers)])
            self.decoder = nn.ModuleList([DecoderLayer(d_model, num_heads) for _ in range(num_layers)])

        def forward(self, src, tgt):
            enc_output = src
            for layer in self.encoder:
                enc_output = layer(enc_output)

            dec_output = tgt
            for layer in self.decoder:
                dec_output = layer(dec_output, enc_output)

            return dec_output
```

Building a generative transformer, while complex, is made accessible with modern libraries and tools. By understanding the steps involved, from data preprocessing to model training and generation, one can harness the power of transformers for a wide range of applications.

3.3. Advanced Techniques and Best Practices

While the foundational concepts and basic implementations provide a solid starting point, mastering generative transformers requires a deeper understanding of advanced techniques and best practices. This section offers insights into improving generation quality, handling long sequences, memory issues, and leveraging fine-tuning and transfer learning [67].

3.3.1. Techniques for Improving Generation Quality

Achieving high-quality text generation necessitates a combination of model architecture tweaks, training strategies, and post-processing methods.

Temperature Sampling. By adjusting the temperature during sampling, one can control the randomness of the generated text [68]. A lower temperature makes the output more deterministic, while a higher value introduces randomness.

$$p_i = \frac{e^{\frac{z_i}{T}}}{\sum_j e^{\frac{z_j}{T}}} \tag{20}$$

where p_i is the adjusted probability, z_i is the original probability, and T is the temperature.

Top-k and Top-p Sampling. Instead of sampling from the entire distribution, one can restrict the sampling pool to the top-k tokens or those tokens that have a cumulative probability greater than a threshold p [69].

Gradient Clipping. To prevent exploding gradients during training, gradient clipping can be employed, ensuring that the gradients remain within a defined range [70]. Gradient clipping can be implemented in PyTorch as follows:

```
torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
```

3.3.2. Handling Long Sequences and Memory Issues

Transformers, by design, have quadratic complexity with respect to sequence length. This can lead to memory issues for long sequences.

Gradient Accumulation. Instead of updating the model weights after every batch, gradients can be accumulated over multiple batches, effectively simulating a larger batch size without the memory overhead [71].

Model Parallelism. For models with billions of parameters, distributing the model across multiple GPUs can alleviate memory constraints [72].

Gradient Checkpointing. This technique involves storing intermediate activations during the forward pass and recomputing them during the backward pass, reducing memory usage at the cost of increased computation.

3.3.3. Fine-Tuning and Transfer Learning

Transfer learning, the practice of leveraging pre-trained models on new tasks, has proven highly effective in the NLP domain.

Fine-tuning. Once a model is pre-trained on a large corpus, it can be fine-tuned on a smaller, task-specific dataset. This approach often yields superior results compared to training from scratch [73,74].

Adapters. Instead of fine-tuning the entire model, adapters allow for training only a small portion of the model, introducing task-specific parameters without altering the pre-trained weights [75].

Mastering generative transformers goes beyond understanding the basics. By incorporating advanced techniques and best practices, one can achieve state-of-the-art performance, handle large models and sequences efficiently, and adapt pre-trained models to new tasks with ease. As the field of NLP continues to evolve, staying abreast of these practices ensures robust and high-quality model deployments.

4. Applications and Use Cases

Generative transformers, with their unparalleled capability to understand and generate human-like text, have found applications across a myriad of domains [40]. This section provides an in-depth exploration of some of the most prominent applications, shedding light on the transformative impact of these models on various industries.

4.1. Text Generation for Creative Writing

The realm of creative writing, traditionally seen as the bastion of human creativity, has witnessed significant advancements with the advent of generative transformers [76]. These models, trained on vast corpora of literature, can produce text that mirrors the style, tone, and complexity of human authors.

Novel and Short Story Generation. AI-powered applications based on GPT-3 and other large language models have been employed to generate entire novels or assist authors by suggesting plot twists, character developments, and dialogues [77]. The generated content, while sometimes requiring human oversight, exhibits creativity and coherence.

Poetry and Song Lyrics. The nuanced and abstract nature of poetry and song lyrics poses a significant challenge for traditional models. However, the advent of generative transformers has enabled these models to produce verses that resonate with human emotions and experiences. A recent study demonstrated that AI-generated poems were often indistinguishable from those written by humans [78], showcasing the success of these algorithms in replicating human-like poetic expressions.

4.2. Chatbots and Conversational Agents

The rise of digital communication has spurred the demand for intelligent chatbots and conversational agents. Generative transformers, with their ability to generate contextually relevant and coherent responses, stand at the forefront of this revolution. One of the most prominent examples of a conversational agent built on generative transformer architecture is ChatGPT, developed by OpenAI. ChatGPT reached 100 million monthly active users just 2 months after launching, making it the fastest-growing application in history.

Customer Support. Businesses employ transformer-based chatbots to handle customer queries, complaints, and feedback [79,80]. These chatbots can understand the context, provide accurate information, and even escalate issues when necessary.

Personal Assistants. Digital personal assistants, such as Siri and Alexa, are integrating transformer models to enhance their conversational capabilities, making interactions more natural and context-aware.

4.3. Code Generation and Programming Assistance

Software development is undergoing a significant transformation with the introduction of transformer models capable of understanding and generating code. One such model that transforms natural language instructions to code is the Codex model developed by OpenAI [81]. These models assist developers by suggesting code snippets, detecting bugs, and even generating entire functions or modules.

Code Completion. Integrated Development Environments (IDEs) are incorporating transformers to provide real-time code completion suggestions, enhancing developer productivity.

Bug Detection and Fixing. Transformers can be trained to detect anomalies in code and suggest potential fixes, reducing debugging time and ensuring more robust software.

4.4. Other Notable Applications

Beyond the aforementioned domains, generative transformers have found applications in diverse areas:

Translation. While traditional machine translation models have limitations, transformers can produce translations that consider the broader context, resulting in more accurate and idiomatic outputs [34].

Summarization. Generative transformers can read lengthy articles or documents and produce concise summaries, retaining the core information and intent [35].

Gaming. In the gaming industry, transformers are used to generate dialogues, plotlines, and even assist in game design by suggesting scenarios or character backstories [82].

The applications of generative transformers are vast and continually expanding. As research progresses and models become more sophisticated, it is anticipated that their integration into various domains will become even more profound.

5. Challenges and Limitations

While generative transformers have showcased remarkable capabilities, they are not devoid of challenges and limitations. This section delves into some of the most pressing concerns surrounding these models, from interpretability issues to ethical dilemmas and computational constraints.

5.1. Model Interpretability

Deep learning models, especially those with millions or billions of parameters such as generative transformers, are often criticized for being "black boxes". Understanding why a model made a particular decision can be elusive [83].

Attention Maps. One approach to interpretability is visualizing attention maps [29,84]. These maps show which parts of the input the model focused on when producing an output. Attention maps are generated by the attention mechanism that computes a set of attention scores, which can be visualized as a heatmap.

Attention maps serve as a tool for interpreting transformer models in NLP by providing insights into various aspects of text processing. They help in analyzing the roles of words in sentences, identifying key topics, evaluating text quality, and detecting errors or biases. However, while attention maps provide insights, they do not offer a complete understanding of the model's decision-making process.

Mathematical Analysis. Efforts are being made to develop mathematical tools and frameworks to dissect the inner workings of transformers [85,86]. Yet, a comprehensive understanding remains a research frontier.

5.2. Hallucination in Text Generation

Generative transformers are sometimes susceptible to generating text that, while coherent and grammatically correct, is factually incorrect or nonsensical. This phenomenon is commonly referred to as a hallucination. Ji et al. conducted a comprehensive survey of the issue of hallucination in natural language generation (NLG) [87].

The causes of hallucination are multifaceted and can vary. They may include inadequate training data, which limits the model's understanding of the subject matter. Overfitting to the training set is another common issue, where the model learns the noise in the data rather than the actual pattern. Additionally, high model complexity leading to over-parameterization can also contribute to hallucination.

Addressing the issue of hallucination involves multiple strategies. One approach is to fine-tune the model on a more specific dataset that is closely aligned with the task at hand. Another strategy involves incorporating external knowledge bases that can fact-check the generated text in real-time. Ensemble methods, which combine the outputs of multiple models, can also be used to validate the generated text and reduce the likelihood of hallucination.

Efforts are underway to quantify the degree of hallucination in generated text. Although a standard measure has yet to be established, one simplistic way to quantify it is through the Hallucination Score, defined as the ratio of the number of hallucinated tokens to the total number of generated tokens, as shown in Equation (21).

$$\text{Hallucination Score} = \frac{\text{Number of hallucinated tokens}}{\text{Total number of generated tokens}} \tag{21}$$

5.3. Ethical Considerations in Text Generation

Generative transformers, with their ability to produce human-like text, raise several ethical concerns [88].

Misinformation and Fake News. There is potential for these models to generate misleading or false information, which can be weaponized to spread misinformation.

Bias and Fairness. Transformers, being trained on vast internet datasets, can inherit and perpetuate biases present in the data [89]. Addressing this requires careful dataset curation and post-hoc bias mitigation techniques.

$$\text{Bias} = \frac{\sum_{i=1}^{n}(P_{\text{model}}(x_i) - P_{\text{true}}(x_i))}{n} \tag{22}$$

where P_{model} is the model's prediction, P_{true} is the true distribution, and n is the number of samples.

5.4. Computational Requirements and Environmental Impact

Training a large language model demands significant computational resources. For example, the GPT-3 model, which has 175 billion parameters, would require 3.14×10^{23} FLOPS for training, translating to 355 GPU-years and a cost of USD 4.6 million on a V100 GPU [90]. Memory is another bottleneck; the model's 175 billion parameters would need 700 GB of memory, far exceeding the capacity of a single GPU. To manage these challenges, OpenAI used model parallelism techniques and trained the models on a high-bandwidth cluster. As language models grow in size, model parallelism is becoming increasingly essential for research.

Energy Consumption. The energy required to train state-of-the-art models can be equivalent to the carbon footprint of multiple car lifetimes. This raises environmental concerns.

Exclusivity. The computational demands mean that only well-funded organizations can train the most advanced models, leading to concerns about the democratization of AI.

While generative transformers offer immense potential, it is crucial to address their challenges and limitations. Balancing the pursuit of state-of-the-art performance with ethical, environmental, and computational considerations is paramount for the sustainable and responsible advancement of the field.

6. The Future of Generative Transformers

Generative transformers, evolving from early models such as the Recurrent Neural Networks (RNNs) to the sophisticated Generative Adversarial Networks (GANs) and now the powerful transformers, have revolutionized numerous domains. With advancements

in model architectures, training techniques, and hardware capabilities, we can anticipate models that not only understand and generate human-like text but also exhibit enhanced creativity, reasoning, and a form of artificial consciousness.

The way forward is full of opportunities for exploration and innovation. As the field of generative transformers continues to evolve, there are numerous avenues for research and development that remain unexplored or underexplored. The evolution from rules-based systems to advanced LLMs has dramatically improved performance and training efficiency. These improvements are not confined to text and language processing but extend to computer vision and other modalities, creating avenues for interdisciplinary research.

6.1. Multimodal Models

The future sees generative models that seamlessly integrate multiple modalities—text, image, sound, video, and more—offering a holistic understanding of the world and generating content that overcomes the limitations of current models. Recent advancements have already led to transformers capable of generating not just text, but also image, audio, and video [91]. These multimodal models are expected to evolve into sophisticated systems capable of processing and understanding inputs from various modalities simultaneously.

In the future, we anticipate the emergence of single applications and more advanced multimodal models. These systems would not only understand inputs from different sensory channels—such as visual, auditory, and textual—but also generate outputs in various forms, moving well beyond mere text generation. The integration of these modalities in a single model offers a more comprehensive approach to understanding complex real-world scenarios and creating more nuanced and contextually relevant outputs.

6.2. Domain-Specific Models

The development of domain-specific GPT models is becoming increasingly crucial across various applications [92]. While current large language models are adept at understanding natural language and generating content, their effectiveness and accuracy can vary significantly when applied to specialized domains such as medicine, law, and finance [93]. A big challenge in tailoring these models to a specific domain lies in the acquisition of high-quality, domain-specific data. Another significant challenge is the fine-tuning process, which involves adapting the model to the unique characteristics and vocabulary of the domain.

Despite these obstacles, there has been progress in the development and implementation of domain-specific GPT models. The emergence of these models marks a future towards more tailored AI solutions. Companies with unique large datasets stand to gain competitive advantages by training their own bespoke models. This trend is exemplified by Bloomberg's development of a specialized LLM for financial tasks [94]. Other companies such as Hugging Face and Databricks are also playing pivotal roles in providing the necessary resources and platforms for developing and fine-tuning these customized models.

In the future, we can expect these domain-specific GPT models to offer enhanced efficiency, improved interpretability, and better domain generability compared to existing large language models. However, the development of these models must also focus on optimizing energy consumption and addressing the challenges of knowledge retention during the fine-tuning process.

6.3. Model Efficiency

The growing size of models necessitates research in computational efficiency and energy consumption. This includes efforts to develop more sustainable AI infrastructure and predictive infrastructure, essential for the data-intensive nature of enterprise AI applications.

6.4. Ethical AI

With the widespread implementation of generative AI across various sectors, ensuring ethical use becomes paramount. This involves research into bias mitigation, fairness, transparency, and the development of guidelines for responsible AI usage [95], especially as AI begins to automate complex tasks like legal work and medical fields like drug design and medical diagnosis.

6.5. Interdisciplinary Integration

The future of generative AI involves its fusion with other fields such as neuroscience and cognitive science. This integration could lead to breakthroughs in understanding both artificial and natural intelligence, with generative AI applications expanding beyond technical fields to impact popular culture and everyday life, such as in the creation of high-resolution images and user-friendly AI applications for enhancing productivity.

7. Conclusions

As we reflect upon the evolution of generative transformers, from their foundational roots with Alan Turing to their current state-of-the-art capabilities, it becomes clear that we are at a turning point in the development of artificial intelligence. In the words of Alan Turing, "We can only see a short distance ahead, but we can see plenty there that needs to be done".

As we reflect upon the evolution of generative transformers, from their foundational roots with Alan Turing to their current state-of-the-art capabilities, it becomes clear that we are at a turning point in the development of artificial intelligence. In the words of Alan Turing, "We can only see a short distance ahead, but we can see plenty there that needs to be done". This foresight aptly describes the current state of AI. The advancements in generative transformers have not only redefined what machines are capable of doing but also opened up a myriad of possibilities for future exploration and innovation. As we advance and develop new technologies, it is crucial to navigate the ethical implications, environmental and societal impacts of these technologies. The goal is not just to push the boundaries of what AI can achieve but to do so responsibly, ensuring that these advancements benefit society at large.

Author Contributions: Conceptualization, investigation, methodology, formal analysis, writing—original draft: E.Y.Z. and A.D.C.; Supervision: Z.P. and J.C.; Writing—review & editing: E.Y.Z., A.D.C., Z.P., J.C. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Research on quality Assurance and Evaluation of higher Education in Jiangsu Province under Grant No. 2023JSETKT032.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Baum, L.E.; Petrie, T. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.* **1966**, *37*, 1554–1563. [CrossRef]
2. Baum, L.E.; Eagon, J.A. An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processes and to a Model for Ecology. 1967. Available online: https://community.ams.org/journals/bull/1967-73-03/S0002-9904-1967-11751-8/S0002-9904-1967-11751-8.pdf (accessed on 10 November 2023).
3. Baum, L.E.; Petrie, T.; Soules, G.; Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.* **1970**, *41*, 164–171. [CrossRef]

4. Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **1989**, *77*, 257–286. [CrossRef]
5. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef] [PubMed]
6. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
7. Kingma, D.P.; Welling, M. An introduction to variational autoencoders. *Found. Trends Mach. Learn.* **2019**, *12*, 307–392. [CrossRef]
8. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
9. Creswell, A.; White, T.; Dumoulin, V.; Arulkumaran, K.; Sengupta, B.; Bharath, A.A. Generative adversarial networks: An overview. *IEEE Signal Process. Mag.* **2018**, *35*, 53–65. [CrossRef]
10. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
11. Antoniou, A.; Storkey, A.; Edwards, H. Data augmentation generative adversarial networks. *arXiv* **2017**, arXiv:1711.04340.
12. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 1–48. [CrossRef]
13. Deecke, L.; Vandermeulen, R.; Ruff, L.; Mandt, S.; Kloft, M. Image anomaly detection with generative adversarial networks. In Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, 10–14 September 2018; Proceedings, Part I 18; Springer: Berlin/Heidelberg, Germany, 2019; pp. 3–17.
14. Yang, Q.; Yan, P.; Zhang, Y.; Yu, H.; Shi, Y.; Mou, X.; Kalra, M.K.; Zhang, Y.; Sun, L.; Wang, G. Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss. *IEEE Trans. Med. Imaging* **2018**, *37*, 1348–1357. [CrossRef] [PubMed]
15. Zhang, H.; Sindagi, V.; Patel, V.M. Image de-raining using a conditional generative adversarial network. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *30*, 3943–3956. [CrossRef]
16. Oord, A.V.D.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv* **2016**, arXiv:1609.03499.
17. Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; Sutskever, I. Zero-shot text-to-image generation. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 8821–8831.
18. Dhariwal, P.; Jun, H.; Payne, C.; Kim, J.W.; Radford, A.; Sutskever, I. Jukebox: A generative model for music. *arXiv* **2020**, arXiv:2005.00341.
19. Cetinic, E.; She, J. Understanding and creating art with AI: Review and outlook. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2022**, *18*, 1–22. [CrossRef]
20. Bian, Y.; Xie, X.Q. Generative chemistry: Drug discovery with deep learning generative models. *J. Mol. Model.* **2021**, *27*, 71. [CrossRef]
21. Stephenson, N.; Shane, E.; Chase, J.; Rowland, J.; Ries, D.; Justice, N.; Zhang, J.; Chan, L.; Cao, R. Survey of machine learning techniques in drug discovery. *Curr. Drug Metab.* **2019**, *20*, 185–193. [CrossRef]
22. Martin, D.; Serrano, A.; Bergman, A.W.; Wetzstein, G.; Masia, B. Scangan360: A generative model of realistic scanpaths for 360 images. *IEEE Trans. Vis. Comput. Graph.* **2022**, *28*, 2003–2013. [CrossRef]
23. Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; Guibas, L. Learning representations and generative models for 3D point clouds. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 40–49.
24. Khoo, E.T.; Lee, S.P.; Cheok, A.D.; Kodagoda, S.; Zhou, Y.; Toh, G.S. Age invaders: Social and physical inter-generational family entertainment. In Proceedings of the CHI'06 Extended Abstracts on Human Factors in Computing Systems, Montreal, QU, Canada, 22–27 April 2006; pp. 243–246.
25. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**, arXiv:1511.06434.
26. Way, G.P.; Greene, C.S. Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders. In Proceedings of the Pacific Symposium on Biocomputing 2018, Hawaii, HI, USA, 3–7 January 2018; World Scientific: Singapore, 2018; pp. 80–91.
27. Sirignano, J.; Cont, R. Universal features of price formation in financial markets: Perspectives from deep learning. *Quant. Financ.* **2019**, *19*, 1449–1459. [CrossRef]
28. Reichstein, M.; Camps-Valls, G.; Stevens, B.; Jung, M.; Denzler, J.; Carvalhais, N.; Prabhat, F. Deep learning and process understanding for data-driven Earth system science. *Nature* **2019**, *566*, 195–204. [CrossRef]
29. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. Available online: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (accessed on 10 November 2023).
30. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Atlanta, GA, USA, 17–19 June 2013; pp. 1310–1318.
31. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
32. Rogers, A.; Kovaleva, O.; Rumshisky, A. A primer in BERTology: What we know about how BERT works. *Trans. Assoc. Comput. Linguist.* **2021**, *8*, 842–866. [CrossRef]

33. Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y.T.; Li, Y.; Lundberg, S.; et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv* **2023**, arXiv:2303.12712.
34. Jiao, W.; Wang, W.; Huang, J.T.; Wang, X.; Tu, Z. Is ChatGPT a good translator? A preliminary study. *arXiv* **2023**, arXiv:2301.08745.
35. Gao, M.; Ruan, J.; Sun, R.; Yin, X.; Yang, S.; Wan, X. Human-like summarization evaluation with chatgpt. *arXiv* **2023**, arXiv:2304.02554.
36. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
37. Raghu, M.; Unterthiner, T.; Kornblith, S.; Zhang, C.; Dosovitskiy, A. Do Vision Transformers See Like Convolutional Neural Networks? *arXiv* **2021**, arXiv:2108.08810.
38. Paul, S.; Chen, P.Y. Vision transformers are robust learners. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2022; Volume 36, pp. 2071–2081.
39. Nikolić, G.S.; Dimitrijević, B.R.; Nikolić, T.R.; Stojcev, M.K. A survey of three types of processing units: CPU, GPU and TPU. In Proceedings of the 2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), Ohrid, Macedonia, 16–18 June 2022; pp. 1–6.
40. Gozalo-Brizuela, R.; Garrido-Merchan, E.C. ChatGPT is not all you need. A State of the Art Review of large Generative AI models. *arXiv* **2023**, arXiv:2301.04655.
41. Lin, T.; Wang, Y.; Liu, X.; Qiu, X. A survey of transformers. *arXiv* **2022**, arXiv:2106.04554.
42. Kalyan, K.S.; Rajasekharan, A.; Sangeetha, S. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv* **2021**, arXiv:2108.05542.
43. Acheampong, F.A.; Nunoo-Mensah, H.; Chen, W. Transformer models for text-based emotion detection: A review of BERT-based approaches. *Artif. Intell. Rev.* **2021**, *54*, 5789–5829. [CrossRef]
44. Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. A survey on vision transformer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 87–110. [CrossRef]
45. Khan, S.; Naseer, M.; Hayat, M.; Zamir, S.W.; Khan, F.S.; Shah, M. Transformers in vision: A survey. *ACM Comput. Surv. (CSUR)* **2022**, *54*, 1–41. [CrossRef]
46. Shamshad, F.; Khan, S.; Zamir, S.W.; Khan, M.H.; Hayat, M.; Khan, F.S.; Fu, H. Transformers in medical imaging: A survey. *Med. Image Anal.* **2023**, *88*, 102802. [CrossRef]
47. Aleissaee, A.A.; Kumar, A.; Anwer, R.M.; Khan, S.; Cholakkal, H.; Xia, G.S.; Khan, F.S. Transformers in remote sensing: A survey. *Remote Sens.* **2023**, *15*, 1860. [CrossRef]
48. Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; Sun, L. Transformers in time series: A survey. *arXiv* **2022**, arXiv:2202.07125.
49. Ahmed, S.; Nielsen, I.E.; Tripathi, A.; Siddiqui, S.; Ramachandran, R.P.; Rasool, G. Transformers in time-series analysis: A tutorial. *Circuits Syst. Signal Process.* **2023**, *42*, 7433–7466. [CrossRef]
50. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. *J. Math* **1936**, *58*, 5.
51. Copeland, B.J. The Church-Turing Thesis. 1997. Available online: https://plato.stanford.edu/ENTRIES/church-turing/ (accessed on 10 November 2023).
52. Bernays, P. Alonzo Church. An unsolvable problem of elementary number theory. *Am. J. Math.* **1936**, *58*, 345–363.
53. Hodges, A. *Alan Turing: The Enigma: The Book That Inspired the Film "The Imitation Game"*; Princeton University Press: Princeton, NJ, USA, 2014.
54. Turing, A.M. *Proposed Electronic Calculator*; National Physical Laboratory: London, UK, 1946.
55. Machinery, C. Computing machinery and intelligence-AM Turing. *Mind* **1950**, *59*, 433.
56. Turing, A. Intelligent machinery (1948). In *The Essential Turing*; Copeland, B.J., Ed.; Oxford Academic: Oxford, UK, 2004; pp. 395–432.
57. Turing, A.M. The chemical basis of morphogenesis. *Philos. Trans. R. Soc. London Ser. Biol. Sci.* **1952**, *237*, 37–72.
58. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
59. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4.
60. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef]
61. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://www.mikecaptain.com/resources/pdf/GPT-1.pdf (accessed on 10 November 2023).
62. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
63. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
64. OpenAI. GPT-4 Technical Report. 2023. Available online: http://xxx.lanl.gov/abs/2303.08774 (accessed on 10 November 2023).
65. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971.
66. Thoppilan, R.; De Freitas, D.; Hall, J.; Shazeer, N.; Kulshreshtha, A.; Cheng, H.T.; Jin, A.; Bos, T.; Baker, L.; Du, Y.; et al. Lamda: Language models for dialog applications. *arXiv* **2022**, arXiv:2201.08239.
67. Zhuang, B.; Liu, J.; Pan, Z.; He, H.; Weng, Y.; Shen, C. A survey on efficient training of transformers. *arXiv* **2023**, arXiv:2302.01107.

68. Xu, F.F.; Alon, U.; Neubig, G.; Hellendoorn, V.J. A systematic evaluation of large language models of code. In Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, New York, NY, USA, 13 June 2022; pp. 1–10.
69. Hewitt, J.; Manning, C.D.; Liang, P. Truncation sampling as language model desmoothing. *arXiv* **2022**, arXiv:2210.15191.
70. Zhang, J.; He, T.; Sra, S.; Jadbabaie, A. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv* **2019**, arXiv:1905.11881.
71. Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W.J. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv* **2017**, arXiv:1712.01887.
72. Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv* **2019**, arXiv:1909.08053.
73. Ziegler, D.M.; Stiennon, N.; Wu, J.; Brown, T.B.; Radford, A.; Amodei, D.; Christiano, P.; Irving, G. Fine-tuning language models from human preferences. *arXiv* **2019**, arXiv:1909.08593.
74. Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; Smith, N. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv* **2020**, arXiv:2002.06305.
75. He, R.; Liu, L.; Ye, H.; Tan, Q.; Ding, B.; Cheng, L.; Low, J.W.; Bing, L.; Si, L. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv* **2021**, arXiv:2106.03164.
76. Shidiq, M. The use of artificial intelligence-based chat-gpt and its challenges for the world of education; from the viewpoint of the development of creative writing skills. In Proceedings of the International Conference on Education, Society and Humanity, Taipei, Taiwan, 28–30 June 2023; Volume 1, pp. 353–357.
77. Ippolito, D.; Yuan, A.; Coenen, A.; Burnam, S. Creative writing with an ai-powered writing assistant: Perspectives from professional writers. *arXiv* **2022**, arXiv:2211.05030.
78. Köbis, N.; Mossink, L.D. Artificial intelligence versus Maya Angelou: Experimental evidence that people cannot differentiate AI-generated from human-written poetry. *Comput. Hum. Behav.* **2021**, *114*, 106553. [CrossRef]
79. Hardalov, M.; Koychev, I.; Nakov, P. Towards automated customer support. In *Artificial Intelligence: Methodology, Systems, and Applications, Proceedings of the 18th International Conference, AIMSA 2018, Varna, Bulgaria, 12–14 September 2018*; Proceedings 18; Springer: Berlin/Heidelberg, Germany, 2018; pp. 48–59.
80. Følstad, A.; Skjuve, M. Chatbots for customer service: User experience and motivation. In Proceedings of the 1st International Conference on Conversational User Interfaces, Dublin, Ireland, 22–23 August 2019; pp. 1–9.
81. Finnie-Ansley, J.; Denny, P.; Becker, B.A.; Luxton-Reilly, A.; Prather, J. The robots are coming: Exploring the implications of openai codex on introductory programming. In Proceedings of the 24th Australasian Computing Education Conference, Melbourne, VIC, Australia, 14–18 February 2022; pp. 10–19.
82. Värtinen, S.; Hämäläinen, P.; Guckelsberger, C. Generating role-playing game quests with gpt language models. *IEEE Trans. Games* **2022**, 1–12. . [CrossRef]
83. Doshi-Velez, F.; Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv* **2017**, arXiv:1702.08608.
84. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 2048–2057.
85. Chefer, H.; Gur, S.; Wolf, L. Transformer interpretability beyond attention visualization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 782–791.
86. Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; et al. A mathematical framework for transformer circuits. *Transform. Circuits Thread* **2021**, *1*. Available online: https://transformer-circuits.pub/2021/framework/index.html (accessed on 10 November 2023).
87. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **2023**, *55*, 1–38. [CrossRef]
88. Ganguli, D.; Hernandez, D.; Lovitt, L.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; Dassarma, N.; Drain, D.; Elhage, N.; et al. Predictability and surprise in large generative models. In Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, 21–24 June 2022; pp. 1747–1764.
89. Silva, A.; Tambwekar, P.; Gombolay, M. Towards a comprehensive understanding and accurate evaluation of societal biases in pre-trained transformers. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 2383–2389.
90. Li, C. OpenAI's GPT-3 Language Model: A Technical Overview. *Lambda Labs Blog* **2020**. Available online: https://lambdalabs.com/blog/demystifying-gpt-3 (accessed on 10 November 2023).
91. Xu, P.; Zhu, X.; Clifton, D.A. Multimodal learning with transformers: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 12113–12132. [CrossRef] [PubMed]
92. Pal, S.; Bhattacharya, M.; Lee, S.S.; Chakraborty, C. A Domain-Specific Next-Generation Large Language Model (LLM) or ChatGPT is Required for Biomedical Engineering and Research. *Ann. Biomed. Eng.* **2023**, 1–4. . [CrossRef]
93. Wang, C.; Liu, X.; Yue, Y.; Tang, X.; Zhang, T.; Jiayang, C.; Yao, Y.; Gao, W.; Hu, X.; Qi, Z.; et al. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv* **2023**, arXiv:2310.07521.

94. Wu, S.; Irsoy, O.; Lu, S.; Dabravolski, V.; Dredze, M.; Gehrmann, S.; Kambadur, P.; Rosenberg, D.; Mann, G. Bloomberggpt: A large language model for finance. *arXiv* **2023**, arXiv:2303.17564.
95. Floridi, L.; Cowls, J.; Beltrametti, M.; Chatila, R.; Chazerand, P.; Dignum, V.; Luetge, C.; Madelin, R.; Pagallo, U.; Rossi, F.; et al. An ethical framework for a good AI society: Opportunities, risks, principles, and recommendations. *Ethics Gov. Policies Artif. Intell.* **2021**, *144*, 19–39.

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