Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Survey of different Large Language Model Architectures: Trends, Benchmarks, and Challenges

MINGHAO SHAO[1], ABDUL BASIT[2], RAMESH KARRI[1], and MUHAMMAD SHAFIQUE[2]
[1]New York University, New York, USA
[2]New York University Abu Dhabi, Abu Dhabi, UAE

Corresponding author: Minghao Shao (e-mail: shao.minghao@nyu.edu).

ABSTRACT Large Language Models (LLMs) represent a class of deep learning models adept at understanding natural language and generating coherent responses to various prompts or queries. These models far exceed the complexity of conventional neural networks, often encompassing dozens of neural network layers and containing billions to trillions of parameters. They are typically trained on vast datasets, utilizing architectures based on transformer blocks. Present-day LLMs are multi-functional, capable of performing a range of tasks from text generation and language translation to question answering, as well as code generation and analysis. An advanced subset of these models, known as Multimodal Large Language Models (MLLMs), extends LLM capabilities to process and interpret multiple data modalities, including images, audio, and video. This enhancement empowers MLLMs with capabilities like video editing, image comprehension, and captioning for visual content. This survey provides a comprehensive overview of the recent advancements in LLMs. We begin by tracing the evolution of LLMs and subsequently delve into the advent and nuances of MLLMs. We analyze emerging state-of-the-art MLLMs, exploring their technical features, strengths, and limitations. Additionally, we present a comparative analysis of these models and discuss their challenges, potential limitations, and prospects for future development.

INDEX TERMS Large Language Models (LLMs), Transformer Architecture, Generative Models, Survey, Multimodal Learning, Deep Learning, Natural Language Processing (NLP).

I. INTRODUCTION

The introduction of the Transformer architecture [1] in 2017 marked an inflection point in the trajectory of Natural Language Processing (NLP) technology. One notable derivative of this innovation is Large Language Models (LLMs). Demonstrating proficiency across multiple NLP tasks, LLMs have been integral for text generation, machine translation, and natural language understanding. Their evolution, spanning several years, has not only underlined their power in linguistic tasks but also showcased their versatility in handling diverse formats like images, videos, and robotic interfaces.

Notable tasks for which LLMs have been used include:

- **Text generation** of coherent text from structured input upon receiving pertinent instructions.
- **Logical reasoning**: Analysis and inference based on logic intrinsic to a given scenario.
- **Machine Translation** across linguistic frameworks.
- **Summarization**: Contextual abridgment of content.
- **Multimodal support**: Beyond textual content, LLMs also facilitate inputs and outputs in various formats, including images, videos, and interactions in robotic environments, leveraging multiple modalities.

The genesis of LLM development can be traced back to 2018 with the advent of GPT [2] and BERT [3]. Each model, crafted with distinct architectures, catered to specific niches within the LLM spectrum. Contemporary LLMs primarily leverage the foundational Transformer architecture and can be grouped as:

- **Auto-encoding**: Primarily encoder-based, these models are tailored for contextual NLP tasks. E.g., BERT and its derivatives.
- **Auto-regressive**: Decoder-centric, these models are optimized for generative tasks. E.g., GPT series.
- **Encoder-Decoder**: Amalgamating both encoder and decoder structures, these LLMs harness the strengths of the preceding two types, albeit with certain trade-offs. E.g., the Pangu series, including Pangu-α and Pangu-Σ [4], [5].

The structure of this paper is designed to furnish a concise yet thorough overview of LLMs. Initially, we delve into the

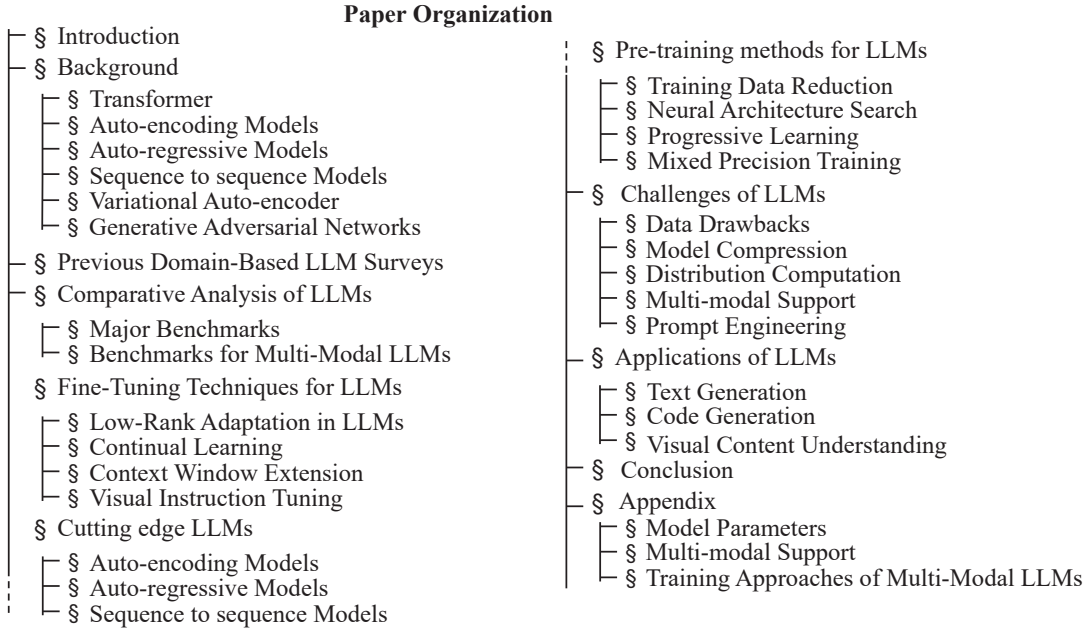

FIGURE 1. Structured layout of the paper is presented, detailing the organization of sections including introduction, background and comparison, state-of-the-art LLMs and methodologies, challenges, and conclusion.

essentials of LLMs, elucidating the underlying technologies and key terminology. We then provide a panoramic view of the LLM evolution, spotlighting influential contributors and institutions. Next, we explore avant-garde LLMs, pivotal in shaping the history of this domain. We culminate with a performance review of LLMs in their designated tasks, concluding with insights and reflections. In essence, our contributions encompass:

1) Constructing a detailed timeline of seminal LLMs up to the release of this paper.
2) Distilling and collating pioneering technologies and strategies pivotal in the evolution of LLMs.
3) Undertaking a holistic comparison of LLMs across architectures and evaluating their performance metrics.
4) Assessing the overarching impacts and challenges posed by contemporary LLMs.

II. BACKGROUND

LLMs predominantly encompass three architectural categorizations: encoder-only, decoder-only, and encoder-decoder. Each category has its unique strengths and constraints and finds relevance across various applications and contexts. This section explains the architecture behind modern LLMs, starting with the general transformer architecture, followed by an exploration of the three categories built upon this architecture.

A. TRANSFORMER

The contemporary landscape of LLMs predominantly employs the Transformer architecture, introduced by Vaswani et al. in 2017 [1]. This architecture represents a paradigm shift away from the recurrent sequence-to-sequence models, such as Long Short-Term Memory (LSTM) networks [6] and Recurrent Neural Networks (RNNs) [7], which were conventionally used. The key innovation of the Transformer lies in its ability to process tokens in parallel, in contrast to the sequential processing constraint in LSTMs and RNNs, where the processing of each token depends on its predecessors. The Transformer achieves this through its multi-head self-attention mechanism, which allows for the parallelized training of models [1].

Conceptually, the Transformer architecture consists of encoder and decoder components. The encoder maps input sequences to a higher-dimensional embedding space, while the decoder generates output sequences from these embeddings. Typically, a Transformer model includes multiple layers of both encoders and decoders. Figure 2 provides an illustrative representation of the Transformer architecture.

Unlike traditional models that process data sequentially, Transformers enable significantly faster and more efficient parallel processing by handling all parts of the input data simultaneously. To address the challenge of maintaining sequence information without inherent sequential processing, Transformers use a technique called **positional encoding**. This mechanism allows each token, such as a word in a sentence, to encode its relative position in the sequence. Positional encoding is essential; without it, the Transformer would treat a sentence as a bag of words, completely oblivious to the order of those words.

Positional encoding utilizes a specific mathematical formula involving sine and cosine functions. This formula ensures that each position in the sequence receives a unique encoding. By appending this encoding to the token's embedding, the model gains insight into the token's position within the sequence. The precise equations used are designed to pro-

Architecture	Advantage	Disadvantage	Example
Auto-Encoding	Good at learning from context, efficient at representation learning	Not suitable for generating sequences	BERT family: BERT, ERNIE, ALBERT, RoBERTa
Auto-Regressive	Suited for generative tasks, effective at language modeling	Lacks context from future tokens during generation	LLaMA family: LLaMA, Alpaca, Vicuna; GPT family
Sequence-to-Sequence	Maps input sequences to word embeddings, conditional generation	High parameter count and complex training	Pangu family: Pangu-α, Pangu-Σ

TABLE 1. Comparison of Auto-Encoding, Auto-Regressive, and Sequence-to-Sequence Models

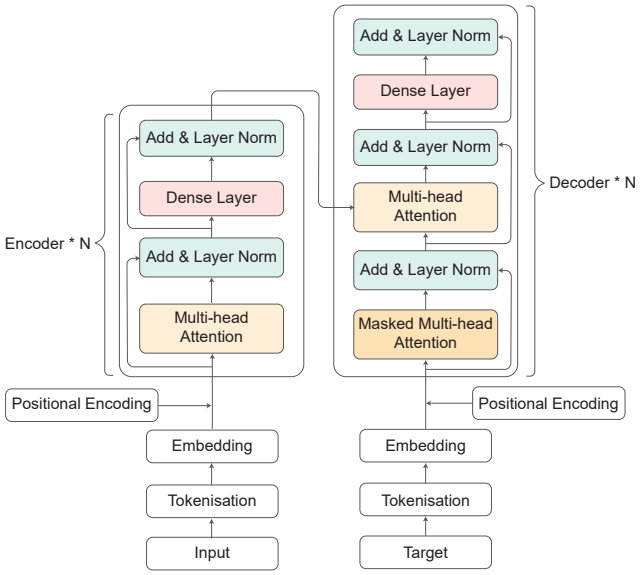

FIGURE 2. The architecture of the Transformer model, which includes an encoder-decoder structure. Key components such as multi-head attention, positional encoding, and residual connections facilitate efficient learning and performance in tasks such as natural language processing and machine translation.

vide a distinct positional signal for every possible position in the input sequence, thus enabling the model to interpret the order of words effectively, despite processing inputs in parallel.

A particular mathematical formula involving the sine and cosine functions is used in positional encoding. This formula ensures that a distinct encoding is assigned to every position in the sequence, enabling the model to be informed about the token's position by appending this encoding to the token's embedding. The precise equations employed are:

$$E(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/\dim}}\right) \quad (1)$$

$$E(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/\dim}}\right) \quad (2)$$

The token's position in the sequence is denoted by pos, while i spans from 0 to half of the embedding dimension (dim), indexing even and odd positions respectively. The choice of sine and cosine functions is particularly advantageous because they provide a unique and consistent way to encode positional information across the embedding space. This setup not only simplifies the model's learning to attend based on relative positions but also enables generalization to sequence lengths beyond those encountered during training. The elegance of this method lies in its ability to imbue the model with the capacity to discern patterns in the data, enriched with positional context. This straightforward yet profound approach has been pivotal to the success of Transformer models in diverse tasks, ranging from text generation and language translation to applications beyond language, such as image recognition.

B. AUTO-ENCODING MODELS

Primarily tailored for natural language processing tasks centered around comprehension, encoder-only models like BERT [3], ERNIE [8], and ALBERT [9] have carved a niche for themselves. Training techniques such as bidirectional learning and masking enable them to excel in contextual understanding. However, they have certain limitations:

- Constrained to fixed-length input sequences.
- Inherent context-dependency can be a hindrance for text generation.
- Given their composition lacks a decoder, downstream task adaptation necessitates fine-tuning.

C. AUTO-REGRESSIVE MODELS

These models, including renowned ones like GPT [2] and the LLaMA series [10], have gained prominence in recent times. Their auto-regressive design implies that token generation hinges on preceding tokens, rendering them apt for generation tasks. These models offer:

- Flexibility in accepting varied input lengths, making them adept at extended data generation.
- Proficiency in few-shot or zero-shot tasks, circumventing the need for specific fine-tuning.
- However, their inability to capture overall context means they draw insights from antecedent tokens.

D. SEQUENCE-TO-SEQUENCE MODELS

Models such as T5 [11] and GLM [12], harmonize strengths of the preceding two types. These models are adept at mapping input sequences to fixed-length embeddings, allowing the decoder to generate contextually relevant outputs. This makes them particularly effective for conditional generation tasks, such as summarization, translation, and question answering, where the output depends closely on the provided input.

The integration of encoder and decoder components enables Seq2Seq models to handle complex inputs but comes with drawbacks:

- Amalgamation increases parameter count, potentially affecting efficiency.
- Training such models requires substantial computational resources due to the complexity of aligning the input and output sequences.

E. VARIATIONAL AUTO-ENCODER

Variational Auto-encoder (VAE) [13] is a sophisticated generative model that evolves from traditional auto-encoders (AE) by integrating probabilistic modeling to develop a meaningful and versatile latent space. Unlike standard AEs, which compress input data into a static representation that the decoder uses to reconstruct the original data, in VAE, the encoder produces a probability distribution defined by means and variances instead of a singular deterministic point. Figure 3 shows the difference between of the principle of auto-encoders and variational auto-encoders.

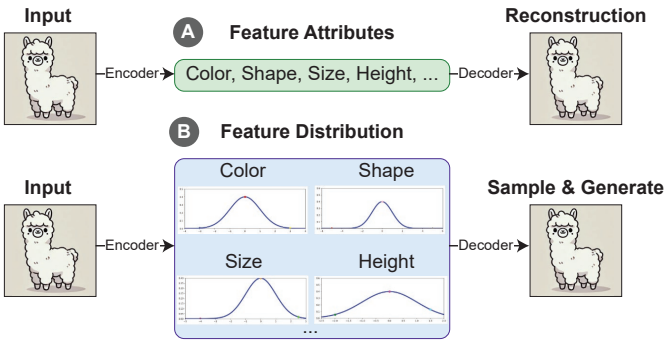

FIGURE 3. (A) Workflow of Auto-encoder, auto-encoder encode the feature attribute directly. (B) Workflow of Variational Auto-encoder, different from auto-encoder, VAEs encode the feature distribution and reconstruct the image based on the sample of distribution, which give the VAEs the ability to generate new images.

VAE utilizes probabilistic encoding to create a dynamic and adaptable latent space, allowing not only data reconstruction but also new data generation by sampling from learned probability distributions. This enhances model generalization and ensures smooth transitions in the latent space, crucial for tasks like data generation and augmentation. It leverages the reparameterization trick to keep gradients flowing through stochastic sampling processes during backpropagation, maintaining the differentiability of latent variables for conventional training. Their objective function balances reconstruction loss, assessing the accuracy of decoded samples against original inputs, with Kullback-Leibler (KL) divergence, promoting approximations of latent distributions to a standard Gaussian. This dual focus ensures both precise input reconstruction and a smooth, continuous latent space, making VAE powerful tools for applications in image generation, data augmentation, and anomaly detection. At the time this paper was released, VAEs had diversified into a wide array of variants. For a comprehensive overview and comparison of these different VAE variants, readers are encouraged to refer to [14] and [15] which provide extensive analyses and insights into the evolution and functionalities of these models.The training of VAE can be represented with following formula

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}[q_\phi(z|x)\|p(z)]$$

Where $\mathbb{E}_{q_\phi(z|x)}[\cdot]$ represents the expectation under the distribution $q_\phi(z|x)$, $\log p_\theta(x|z)$ is the logarithm of the likelihood, which is how well the model can reconstruct the input from the latent variables, $\mathrm{KL}[q_\phi(z|x)\|p(z)]$ is the KL divergence between the approximate posterior $q_\phi(z|x)$ as a regularization by encouraging the posterior to be close to the prior with a Gaussian distribution.

F. GENERATIVE ADVERSARIAL NETWORK

Generative Adversarial Networks (GANs) are a class of DL frameworks introduced by Goodfellow et al. [16]. GANs consist of two neural networks, a generator and a discriminator, which are trained simultaneously through adversarial processes. The generator aims to create synthetic data that resembles the real data, while the discriminator's role is to distinguish between real and synthetic data. Over time, as training progresses, the generator becomes better at creating realistic data, and the discriminator becomes better at differentiating real from fake data, illustrated in Figure 4.

- **Generator**: Takes a noise vector (randomly sampled) and maps it into a data space, aiming to produce samples that resemble the training data. The generator's goal is to generate data that is indistinguishable from the real data.
- **Discriminator**: Takes a sample (real or generated) and predicts whether it is real (from the training dataset) or fake (generated by the generator). It is trained to maximize the probability of correctly classifying real and generated samples.

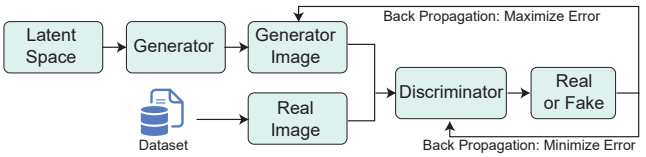

FIGURE 4. Basic architecture of a Generative Adversarial Network (GAN). The generator creates synthetic images from a latent space, while the discriminator distinguishes between real images from the dataset and generated images. The generator is trained to maximize the discriminator's error, while the discriminator is trained to minimize its error in distinguishing real from fake images, leading to adversarial learning between the two components.

The training process of GANs can be described as a min-max game where G is the Generator, D is the Discriminator, x represents real data samples, z is a noise vector, p_{data} denotes the data distribution, and p_z is the noise distribution.

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

1) Variants of GANs

Since their inception, numerous variations of GANs have been proposed to address specific challenges such as mode

GAN Model	Year	Key Idea
Vanilla GAN [16]	2014	Introduced the fundamental adversarial training between the generator and discriminator.
DCGAN [17]	2015	Utilizes convolutional layers to enhance the performance and stability of GANs in image generation tasks.
CGAN [18]	2014	Introduces conditioning variables (e.g., class labels) into both the generator and discriminator to control the output.
WGAN [19]	2017	Replaces the original GAN loss with the Wasserstein distance to improve training stability and reduce mode collapse.
WGAN-GP [20]	2017	Extends WGAN by adding a gradient penalty term to enforce the Lipschitz constraint more effectively.
LSGAN [21]	2017	Uses least-squares loss instead of the cross-entropy loss to address vanishing gradients and stabilize training.
CycleGAN [22]	2017	Introduces cycle consistency loss to enable image-to-image translation without paired training data.
StyleGAN [23]	2019	Introduces a style-based generator architecture, allowing control over different aspects and details of generated images.
BigGAN [24]	2018	Focuses on scaling up GANs using large batch sizes and deeper architectures to generate higher-quality images.
SAGAN [25]	2018	Incorporates self-attention mechanisms in GANs to capture long-range dependencies and generate detailed images.
Progressive GAN [26]	2017	Gradually increases the resolution of generated images during training to achieve more stable results.
StarGAN [27]	2018	Aims to perform multi-domain image-to-image translation using a single generator and discriminator.

TABLE 2. Summary of Different GAN Models and Their Key Ideas

collapse, training stability, and applicability to different types of data (e.g., images, text, or audio). Table 2 contains the summary of some of the most widely used GAN variants along with their unique characteristics.

Several surveys have extensively covered the advancements and applications of GANs. Wang et al. [28] provide a detailed overview of GAN architectures and challenges like mode collapse. Gui et al. [29] review methods to stabilize training and improve image quality, while Creswell et al. [30] focus on GANs' creative applications, including style transfer. Given the depth of these reviews, our paper will concentrate on Large Language Models (LLMs), exploring their generative capabilities and contributions to natural language understanding.

III. PREVIOUS DOMAIN-BASED LLM SURVEYS

In this section, we conduct a comprehensive analysis of the existing surveys on large language models (LLMs). We provide a comparative evaluation of these survey papers based on the topics they address. The surveys are chronologically organized in Table 3, allowing readers to track the evolution of research focus over time. By examining the content covered in these surveys, as summarized in the table, readers can gain a nuanced understanding of the progress made in the development of advanced LLMs. The categories are comprised of:

- **Architecture:** Details about the structural design of the LLMs discussed, including model types and configurations, including Decoder only, Encoder only, and Decoder-Encoder Models.
- **Dataset:** Information about the datasets used for training and evaluating the LLMs.
- **Pre-training:** Methods and techniques used for training the foundation LLMs.
- **Fine-tuning:** Strategies for adapting pre-trained LLMs to specific tasks or domains to improve domain-specific performance.
- **Benchmark:** Evaluation metrics and benchmark datasets used to assess LLM/ MLLM performance.
- **Challenges:** Identification of challenges and techniques to optimize development and deployment of LLMs.
- **MLLMs:** Discussion on Multilingual Language Models and their specific considerations.
- **Applications:** Real-world applications and use cases of state-of-the-art LLMs.

Our survey provides a brief overview of all these categories but delves deeply into the Architecture, Benchmark, and Challenges aspects. This in-depth focus aims to offer a detailed understanding of the structural innovations, evaluation methodologies, and challenges in the field of large language models. Some notable surveys in the domain of LLM/MLLMs are highlighted:

- **The survey by Xiao et al. [40]** provides an extensive overview of LLMs and MLLMs within the medical domain. However, our survey addresses the general methodologies for fine-tuning and LLM architectures, applicable to a wide range of use cases beyond the medical field. By addressing these broader aspects, our survey provides a more holistic view of the advancements, challenges, and future directions in the field of LLMs, making it a valuable resource for a wider audience.
- **The survey by Yin et al. [49]** offers an overview of the progress in multimodal large language models (MLLMs). It covers the basic formulation, related concepts, research topics, technical points, challenges, and future directions of MLLMs. While the MLLM survey provides valuable insights, our survey fills the gaps by offering detailed technical analysis, broader domain coverage, comprehensive comparative evaluations, and in-depth discussions on challenges. Our survey is a more versatile and informative resource for a wider audience in regards to MLLMs.
- **The notable survey from Zhao et al. [81]** from early 2023 offers a comprehensive review of the development and applications of LLMs. However, this survey does not cover the latest developments and models in the fast-evolving field of LLMs. Since the pace of advancement in LLMs is rapid, many newer models and techniques that have emerged in the latter part of 2023 and early 2024 are not included. Our survey fills this gap by including the most recent advancements and models, and providing an in-depth benchmarking analysis for researchers and practitioners.

Table 4 presents a review of the available survey papers on

TABLE 3. Survey Papers on Large Language Models: A comparative analysis

LLM Survey	Month	Year	# Refs.	Architecture	Dataset	Pre-training	Fine-tuning	Benchmark	Challenges	MLLMs	Applications
Ours	Jul	2024	427	✓	✓	✓	✓	✓	✓	✓	✓
[31]	Jul	2024	260	✗	✓	✗	✓	✓	✓	✗	✗
[32]	Jun	2024	392	✓	✗	✓	✓	✓	✓	✓	✗
[33]	Jun	2024	299	✗	✗	✓	✓	✓	✓	✗	✓
[34]	Jun	2024	233	✗	✓	✗	✓	✓	✓	✗	✓
[35]	Jun	2024	221	✓	✗	✗	✗	✗	✓	✗	✓
[36]	Jun	2024	154	✗	✓	✗	✓	✓	✓	✓	✗
[37]	Jun	2024	129	✗	✗	✓	✓	✗	✗	✗	✓
[38]	Jun	2024	42	✗	✓	✓	✓	✓	✓	✗	✓
[39]	May	2024	336	✗	✓	✓	✓	✓	✓	✓	✗
[40]	May	2024	269	✓	✓	✓	✓	✓	✓	✓	✓
[41]	May	2024	207	✓	✓	✓	✓	✗	✓	✓	✓
[42]	May	2024	176	✗	✗	✗	✗	✓	✓	✗	✓
[43]	May	2024	171	✗	✗	✗	✗	✓	✓	✗	✓
[44]	May	2024	168	✗	✗	✓	✓	✗	✓	✗	✓
[45]	May	2024	84	✗	✗	✗	✗	✗	✓	✗	✓
[46]	May	2024	42	✓	✓	✓	✗	✗	✗	✗	✓
[47]	Apr	2024	349	✗	✓	✓	✓	✗	✗	✗	✓
[48]	Apr	2024	208	✗	✓	✗	✓	✓	✓	✗	✗
[49]	Apr	2024	206	✓	✓	✓	✓	✓	✓	✓	✓
[50]	Apr	2024	174	✗	✗	✗	✓	✓	✓	✗	✓
[51]	Apr	2024	170	✗	✗	✗	✓	✓	✓	✓	✓
[52]	Apr	2024	140	✗	✓	✓	✓	✓	✓	✓	✗
[53]	Apr	2024	108	✗	✓	✓	✓	✗	✓	✗	✓
[54]	Apr	2024	54	✓	✗	✗	✓	✗	✗	✗	✓
[55]	Mar	2024	471	✗	✓	✓	✗	✓	✓	✗	✓
[56]	Mar	2024	353	✗	✗	✗	✓	✗	✓	✗	✓
[57]	Mar	2024	269	✗	✓	✗	✗	✓	✓	✗	✓
[58]	Mar	2024	182	✗	✓	✓	✓	✓	✓	✗	✓
[59]	Mar	2024	165	✓	✓	✗	✓	✓	✗	✓	✗
[60]	Mar	2024	148	✗	✓	✗	✓	✓	✓	✗	✗
[61]	Mar	2024	140	✗	✗	✓	✓	✗	✓	✗	✗
[62]	Mar	2024	128	✓	✗	✗	✓	✓	✓	✗	✓
[63]	Mar	2024	127	✗	✓	✗	✗	✓	✓	✓	✓
[64]	Mar	2024	121	✓	✗	✗	✗	✗	✓	✗	✓
[65]	Mar	2024	66	✗	✓	✗	✗	✓	✓	✗	✗
[66]	Mar	2024	25	✗	✓	✗	✗	✗	✓	✗	✓
[67]	Feb	2024	218	✗	✗	✗	✓	✓	✓	✗	✓
[68]	Jan	2024	231	✓	✓	✓	✓	✓	✓	✗	✓
[69]	Jan	2024	46	✗	✓	✗	✗	✓	✗	✗	✗
[70]	Dec	2023	675	✗	✓	✓	✓	✗	✓	✗	✓
[71]	Nov	2023	268	✗	✓	✓	✓	✓	✓	✓	✗
[72]	Oct	2023	285	✓	✓	✓	✓	✗	✗	✓	✓
[73]	Oct	2023	172	✓	✗	✗	✓	✗	✓	✓	✓
[74]	Sep	2023	362	✓	✓	✗	✓	✗	✗	✗	✓
[75]	Sep	2023	215	✗	✓	✓	✓	✓	✓	✓	✓
[76]	Sep	2023	132	✗	✓	✗	✓	✓	✓	✗	✗
[77]	Aug	2023	109	✓	✗	✓	✓	✗	✓	✓	✗
[78]	Jul	2023	378	✓	✓	✓	✓	✓	✗	✓	✓
[79]	May	2023	131	✗	✓	✗	✓	✓	✓	✗	✗
[80]	Apr	2023	65	✗	✗	✗	✗	✗	✓	✗	✓
[81]	Mar	2023	946	✓	✓	✓	✓	✓	✓	✓	✓
[82]	Feb	2023	191	✗	✗	✓	✓	✓	✓	✗	✗
[83]	Feb	2022	216	✓	✗	✗	✓	✓	✓	✗	✓
[84]	Aug	2021	304	✓	✓	✓	✓	✓	✓	✗	✗
[85]	Feb	2020	20	✓	✗	✓	✓	✓	✓	✗	✗

large language models (LLMs) as of the time of this writing. Each survey offers unique insights into the application of LLMs across various domains. To facilitate a clearer understanding, these surveys are categorized into three types:

- General surveys providing an overview of the evolution of LLMs.
- Application-oriented surveys focusing on specific fields.
- Technical surveys detailing the algorithms and techniques used in modern LLMs.

This comprehensive categorization aids in understanding how LLMs adapt to specialized vocabularies and regulatory frameworks across different settings.

Overall, our survey on LLMs serve as a valuable resource for researchers and practitioners, offering a consolidated view of the state-of-the-art in LLMs.

IV. COMPARATIVE ANALYSIS OF LLMS

This section provides a comparative analysis of prominent Language Models using various benchmarks, which assess the models' capabilities in language understanding, reasoning, and multimodal tasks. These benchmarks are designed to evaluate different aspects of language comprehension and

TABLE 4. Collection of previous LLM survey paper categorized into the major topic of their discussion

Category	Field	Literature
Overview	General LLMs	[44], [46], [74], [81], [84]
	Multilingual	[47]
	Multimodality	[39], [49]
Technology	Evaluation	[57]
	Agent Systems	[50], [51], [70]
	Explainability & Reasoning	[60], [67], [79]
	Instruction Tuning	[59]
	Model Compression	[76]
	Hallucination	[52], [71], [75]
	Fast Inference	[33]
	Augmentation	[53], [58], [77], [82]
	Alignment	[55]
	Tool Usage	[43]
	Continual Learning	[32]
	Controllable Generation	[73]
	Security & Privacy	[31], [35], [36], [48], [56], [61], [64], [65], [69], [72]
Application	Casual Inference	[63]
	Information Retrieval	[68]
	Education	[80]
	Reinforcement Learning	[62]
	Legal	[66]
	Recommendation System	[37]
	Telecommunication	[41]
	Conversational AI	[85]
	Text Generation	[54]
	Medical & Biological	[40], [42], [78], [83]
	Transportation	[45]
	Data Science	[34]
	Security	[38]

cognitive abilities.

A. MAJOR BENCHMARKS

Standardized benchmarks are essential for evaluating LLMs' performance across tasks, offering a means to compare models and identify their strengths and weaknesses. This section highlights some of the most widely used benchmarks.

1) MMLU (Massive Multitask Language Understanding)

The Massive Multitask Language Understanding (MMLU) benchmark is a collection of 57 tasks that cover a wide range of subjects from human-level concepts to high school examinations. This benchmark evaluates the comprehensive understanding and generalization power of language models across diverse topics.

Key Features:

- **Coverage:** Broad domain coverage from humanities to STEM.
- **Task Types:** Multiple choice questions with four options, requiring not only language understanding but also domain-specific knowledge.
- **Evaluation Metric:** Accuracy of predicting the correct answer among the given choices.

2) SuperGLUE

SuperGLUE is designed as an advanced benchmark to evaluate and promote improvements in the critical reasoning and prediction-making abilities of AI models beyond the

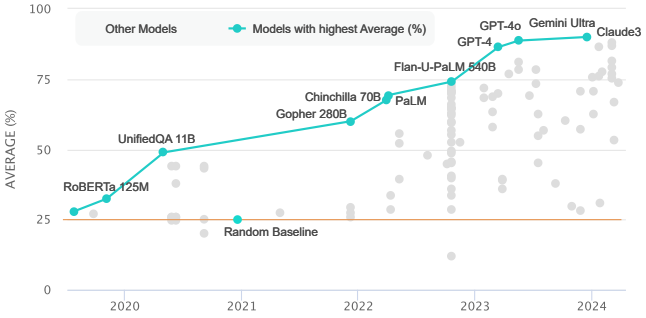

FIGURE 5. Performance trajectory of various LLMs on the MMLU benchmark illustrating average accuracy percentages. The 'Random Baseline' represents a lower bound of performance, while the highlighted teal line traces the models with the highest average scores each year, culminating with the introduction of models like GPT-4o and Gemini Ultra that set new benchmarks for language understanding.

GLUE benchmark. It includes a set of more challenging tasks that require deeper natural language understanding and broader reasoning capabilities.

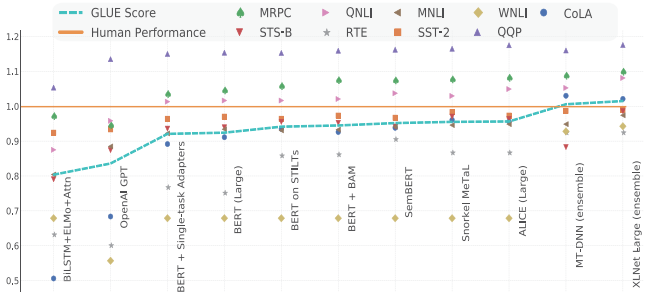

FIGURE 6. Comparative performance visualization of models on the GLUE benchmark, adjusted to a unified scale with human performance normalized to a score of 1.0. The summary score represents an aggregate of nine individual tasks, with an averaged score for tasks containing multiple metrics. The breakdown across the tasks demonstrates the relative strengths and weaknesses of each submitted system in various areas of language understanding.

Key Features:

- **Complexity:** Tasks are specifically chosen to be more difficult and diverse than those in GLUE.
- **Task Variety:** Includes question answering, entailment, coreference resolution, and word sense disambiguation, among others.
- **Data Sources:** Comprises datasets that have been either newly created or significantly expanded upon for heightened difficulty and variability.
- **Evaluation Metric:** Composite score that is calculated based on performance across all constituent tasks, promoting models that achieve balanced capabilities across a broader array of challenges.

3) HellaSwag

HellaSwag is a benchmark designed to test a model's common sense and ability to complete scenarios using everyday knowledge. The tasks involve predicting the ending of descriptions of everyday activities.

Key Features:

- **Contexts:** Comes from diverse sources such as Wikipedia and instructional videos.
- **Task Type:** Choose the most plausible continuation among four provided options.
- **Evaluation Metric:** Accuracy of predictions.

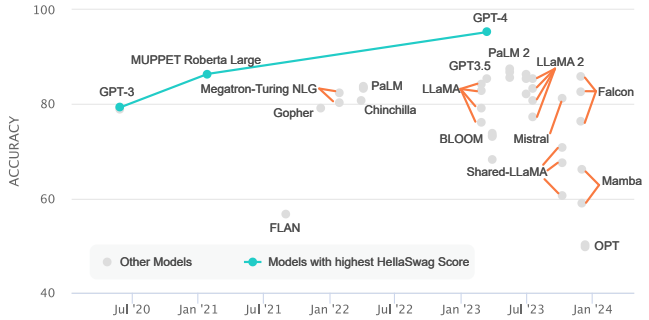

FIGURE 7. Progression of HellaSwag benchmark accuracy scores from tracks the performance of various LLMs achieving the highest HellaSwag scores. Notable milestones include the introduction of models like GPT-4 and PaLM 2, which significantly surpass previous models in accuracy, indicating substantial advancements in AI's commonsense reasoning and contextual understanding abilities.

4) ARC (AI2 Reasoning Challenge)

ARC presents models with grade-school-level multiple-choice science questions, testing their ability to understand text and apply reasoning skills.

Key Features:

- **Difficulty Levels:** Contains both Easy and Challenge sets to adapt to different model capabilities.
- **Evaluation Metric:** Accuracy on correctly answered questions.

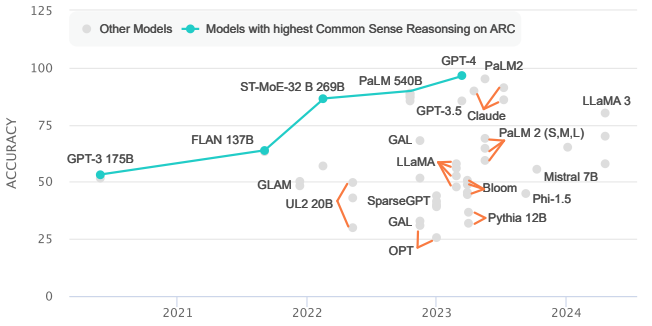

FIGURE 8. Advancement in accuracy of various LLMs on the AI2 Reasoning Challenge (ARC) aimed at appraising common sense reasoning. Newer models like GPT-4 and PaLM demonstrate significantly improved common sense reasoning capabilities. Notably, variations in model training approaches, such as zero-shot and few-shot learning, are reflected in differentiated performance levels.

5) WinoGrande

WinoGrande is a large-scale dataset of winograd schemas that are designed to test common sense reasoning within AI models.

Key Features:

- **Scale:** One of the largest datasets for commonsense reasoning.
- **Task Type:** Sentence completion that requires resolving ambiguous pronouns.
- **Evaluation Metric:** Accuracy of choosing the correct entity.

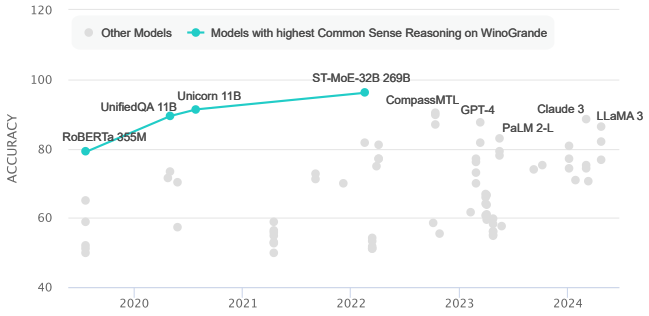

FIGURE 9. Showcasing the ascent in accuracy of diverse LLMs on the WinoGrande benchmark. Remarkable progress is seen with the likes of GPT-4 and the various iterations of the PaLM model, reflecting significant advancements in the field's pursuit of nuanced language understanding and common sense inference capabilities.

B. BENCHMARKS FOR MULTIMODAL LLMS

As the field of AI progresses, benchmarks for evaluating multimodal capabilities of LLMs have become increasingly relevant. Some notable multimodal benchmarks include:

1) NLVR2 (Natural Language for Visual Reasoning for Real)

The NLVR2 benchmark is a challenging dataset designed for evaluating AI models' ability in visual reasoning with natural language. It requires models to determine whether a given natural language statement accurately describes a pair of images. Unlike standard object recognition or image captioning tasks, NLVR2 demands a deeper understanding of both the visual content and the semantics of the language, making it a more complex challenge.

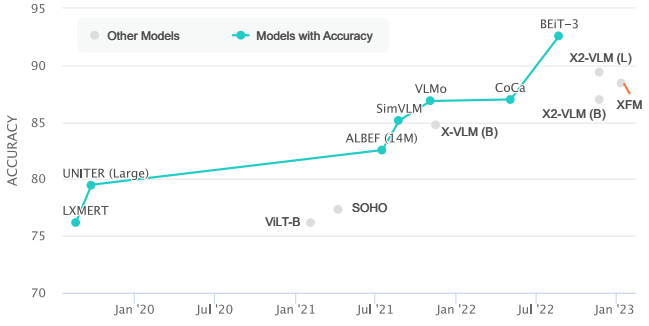

FIGURE 10. This graph depicts the accuracy trend of models on the NLVR2 benchmark. The multi-modal LLMs showcase rapid improvements in visual and linguistic reasoning capabilities. Notable performers such as BEiT-3 and X2-VLM (L) represent the cutting edge of multi-modal LLMs, indicating their superior proficiency in interpreting complex visual-language tasks.

Key Features:

- **Data Composition:** Consists of pairs of images accompanied by a textual description. The model's task

is to verify the truthfulness of the description given the image pair.

- **Reasoning Requirement:** The models must interpret the images in the context of spatial relations, counting, and comparison, making this benchmark particularly suited for evaluating multimodal comprehension.
- **Performance Metrics:** The main metric is accuracy, indicating the model's ability to correctly validate the statement against the visuals.
- **Challenges:** Involves disambiguating ambiguous language, understanding complex statements, and a deep integration of visual and textual information.

Models successful on NLVR2 must not only integrate visual and textual information but also accurately capture the subtleties of language that relate to the visual world.

2) Visual Question Answering (VQA) Benchmark

The Visual Question Answering (VQA) benchmark stands as a measure of an AI system's ability to answer questions pertaining to given images. This multimodal benchmark combines natural language processing with image recognition to test a model's comprehensive understanding of visual content as it relates to conceptual and factual queries.

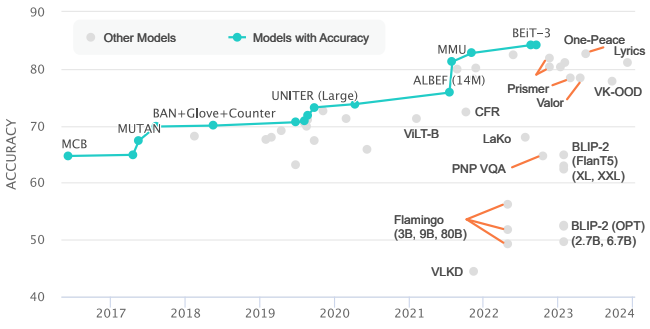

FIGURE 11. Accuracy performance graph for various AI models on the VQA, illustrating the evolution of AI in comprehending and answering visually grounded questions. The graph underscores the significant strides made in multimodal AI, with models like BEiT-3 showing remarkable precision in visual-textual understanding.

Key Features:

- **Task Composition:** Involves an open-ended task where a model is presented with an image and a related question in natural language. The model must provide an accurate answer to the question, reflecting a correct understanding of the visual context.
- **Diversity of Questions:** The questions are designed to cover a wide array of types, including object detection, counting, color determination, spatial understanding, and inferential reasoning based on the image content.
- **Answer Formats:** The answers may be in various forms, including single words, numbers, or short phrases.
- **Performance Metrics:** Accuracy is the primary metric, supplemented by consistency and plausibility scores in some variations of the benchmark.
- **Challenges:** The task challenges models to understand and process visual data in conjunction with textual information, requiring a high level of multimodal integration and reasoning.

The VQA benchmark is crucial for the development of AI systems that interact with the visual world in a meaningful and contextually aware manner, a necessity for applications ranging from assistive technologies to automated content moderation.

These benchmarks test the integration of visual and textual data, crucial for applications requiring a holistic understanding of multimodal inputs.

V. FINE-TUNING TECHNIQUES FOR LLMS

Fine-tuning methods for LLMs are utilized in a variety of applications including domain specialization, performance improvement, and bias mitigation. Key approaches to fine-tuning LLMs, such as Parameter-Efficient Fine-Tuning (PEFT), are often emphasized due to their diverse applications and reduced computational demands as compared to complete model training and these techniques are detailed in this section.

A. LOW-RANK ADAPTATION IN LLMS

Low-Rank Adaptation (LoRA) [157], and specifically Low-Rank-parametrized Update Matrices, provides an efficient strategy for fine-tuning pre-trained Transformer-based language models. This technique is articulated by the following update rule:

$$\Delta W = BA \tag{3}$$

where $W_0 \in \mathbb{R}^{d\times k}$ is the original weight matrix, and ΔW is the low-rank update represented by matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with the rank $r \ll \min(d, k)$. During adaptation, W_0 remains unchanged, and only B and A are trained.

LoRA posits a reduction in the number of trainable parameters, significantly reducing computational overhead. It also generalizes full fine-tuning, theoretically allowing a model to approximate the expressiveness of full-rank weight matrices by selecting an appropriate r. For new tasks, one can quickly adapt the base model W_0 by adjusting BA, thus avoiding additional inference latency.

Furthermore, LoRA introduces a scaling parameter α in the adaptation step:

$$\Delta Wx = \alpha \frac{\Delta Wx}{r} \tag{4}$$

which helps maintain stability in the learning rate when varying the rank r, reducing the necessity for hyperparameter retuning. This parameter-efficient method for adapting LLMs presents an avenue for tailoring models to specialized domains or tasks without forfeiting their original capabilities.

B. CONTINUAL LEARNING

Continual Learning (CL) with PEFT for LLMs is an approach that focuses on adapting a model to new tasks over time while avoiding catastrophic forgetting of previously learned information. It leverages PEFT methods to introduce minimal,

PEFT Methods for PLMs	Subcategory	Techniques
Additive Fine-tuning	Adapter-based Fine-tuning	Adapter Design: Serial Adapter [86], Parallel Adapter [87], CIAT [88], CoDA [89] Multi-task Adaptation: AdapterFusion [90], AdaMix [91], PHA [92], AdapterSoup [93], MerA [94], Hyperformer [95]
	Soft Prompt-based Fine-tuning	Soft Prompt Design: Prefix-tuning [96], Prefix-Propagation [97], p-tuning v2 [98], APT [99], p-tuning [100], prompt-tuning [101], Xprompt [102], IDPG [103], LPT [104], SPT [105], APrompt [106] Training Speedup: SPoT [107], TPT [108], InfoPrompt [109], PTP [110], IPT [111], SMoP [112], DePT [113]
	Others	$(IA)^3$ [114], MoV [115], SSF [116], IPA [117]
Selective Fine-tuning	Unstructural Masking	U-Diff pruning [118], U-BitFit [119], PaFi [120], FishMask [121], Fish-Dip [122], LT-SFT [123], SAM [124], Child-tuning [125]
	Structural Masking	S-Diff pruning, S-BitFit, FAR [126], BitFit [127], Xattn Tuning [128], SPT [105]
Reparameterized Fine-tuning	Low-rank Decomposition	Intrinsic SAID [129], LoRA [130], Compacter [131], KronA [132], KAdaptation [133], HiWi, VeRA [134], DoRA [135] Dynamic Rank: DyLoRA [136], AdaLoRA [137], SoRA [138], CapaBoost [139], AutoLoRA [140]
	LoRA Derivatives	LoRA Improvement: Laplace-LoRA [141], LoRA Dropout [142], PeriodicLoRA [143], LoRA+ [144], LongLoRA [145] Multiple LoRA: LoRAHub [146], MOELoRA [147], MoLoRA, MoLA [148], MoLE [149], MixLoRA [150]
Hybrid Fine-tuning	-	UniPELT [151], S4 [152], MAM Adapter, NOAH [153], AUTOPET [154], LLM-Adapters [155], S^3PET [156]

TABLE 5. Parameter Efficient Fine-tuning methods for Pre-trained Language Models

task-specific updates to the model's parameters. Techniques such as AdapterCL use residual adapters to encapsulate new knowledge for each task. These strategies help maintain the model's performance across a sequence of tasks by incorporating mechanisms like entropy-based classifiers for adapter selection, and by employing strategies to ensure knowledge transfer between tasks. The goal is to achieve a balance where the model continually accumulates and refines knowledge without substantial loss of prior learning.

C. CONTEXT WINDOW EXTENSION

Context Window Extension in PEFT refers to the adaptation of LLMs to process input sequences that exceed their initially defined context lengths. Through PEFT, such as LongLoRA [145], LLMs can be efficiently fine-tuned to extend their context windows, allowing them to handle longer input sequences without a significant increase in computational requirements. This is particularly useful for tasks where the ability to maintain longer context is crucial for performance. LongLoRA and similar techniques modify attention mechanisms and introduce sparse attention patterns to manage longer sequences, enhancing the model's applicability to real-world scenarios with lengthy textual data.

D. VISUAL INSTRUCTION TUNING

One notable PEFT technique is visual instruction tuning, where LLMs, traditionally text-based, are adapted to handle visual inputs, enabling them to perform tasks like image captioning and visual question answering. The integration of visual and language processing in LLMs through visual instruction tuning represents a significant leap in multimodal AI capabilities. The process involves using LLMs like GPT-4 to generate language-image instruction-following data, which is then used to fine-tune a model capable of understanding and interacting with both textual and visual inputs. The resulting model, dubbed LLaVA (Large Language and Vision Assistant) [158], showcases impressive multimodal conversational abilities and has set new benchmarks in accuracy for tasks such as Science QA [159]. This approach underscores the potential of LLMs in general-purpose visual and language understanding tasks. Alternatively, PEFT approaches such as adapter modules are employed to refine models like VL-BART [160] for image-text tasks more efficiently.

These methods represent only a fraction of the PEFT techniques used to adapt LLMs for specialized applications, highlighting the field's adaptability and ongoing innovation. Table 5 offers an in-depth categorization of these PEFT methods for LLMs, showcasing their variety and application potential in NLP.

VI. CUTTING EDGE LLMS

In the following section, we provide an overview of large language models (LLMs) based on their architecture and the series they belong to, as of the date of this survey. This will offer a comprehensive understanding of the various LLMs and their respective design frameworks. Figure 12 illustrates the evolutionary tree showing the progression of LLMs across different architectures. Figure 13 presents a plot of parameter counts for prominent LLMs, highlighting the trend of increasing parameter sizes over the years. Similarly, Figure 14 displays the average scores from the Open-LLM Leaderboard for various benchmarks, including MMLU, ARC, HellaSwag, and TruthfulQA.

A. AUTO-ENCODING MODELS

Auto-encoding models, often referred to as 'encoder-only models,' utilize solely the encoder component of the Transformer architecture. Their primary function is to map input data from a higher-dimensional space to a lower-dimensional vector space, effectively capturing and integrating contextual

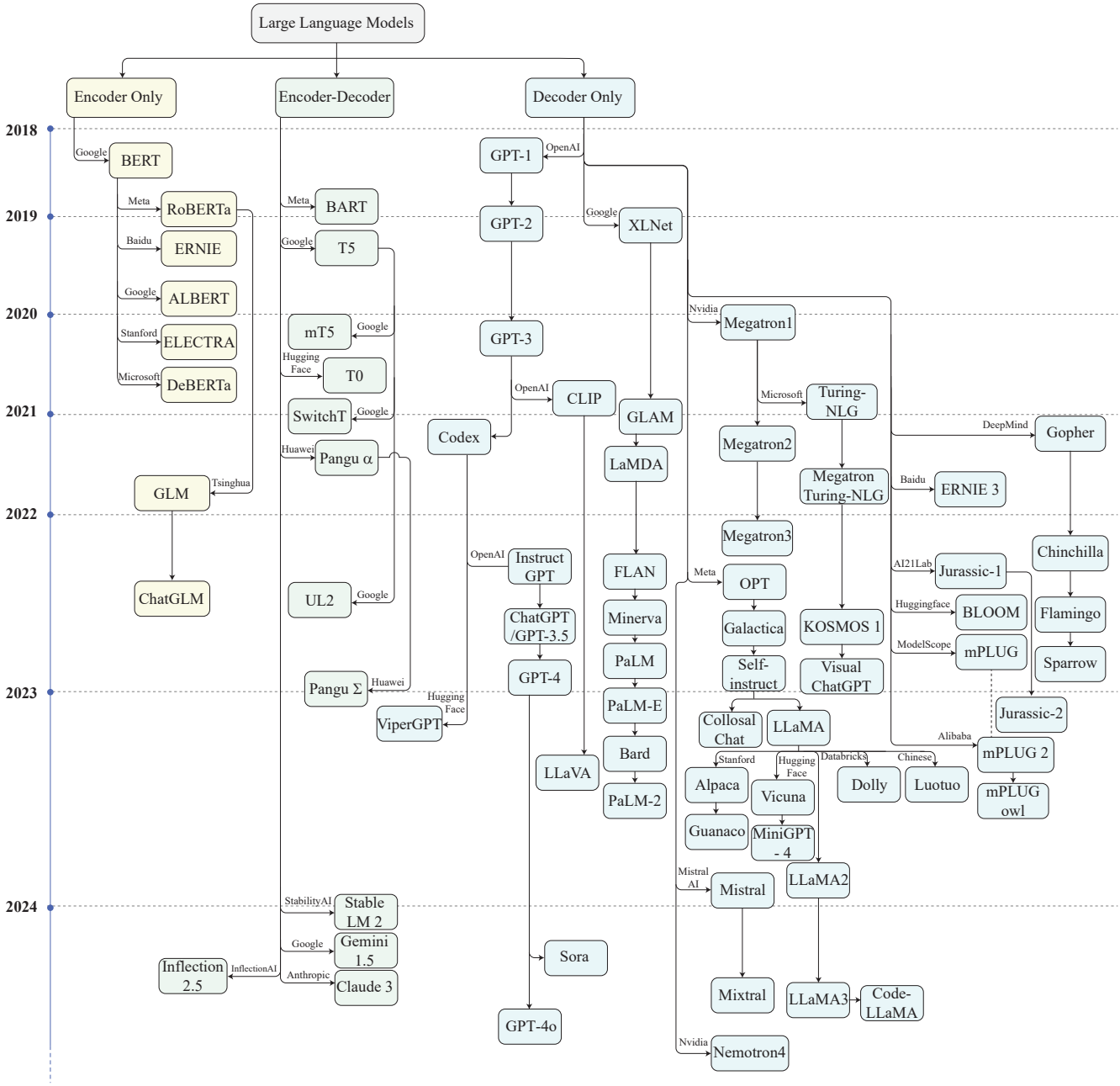

FIGURE 12. An evolutionary tree illustrating the progression of mainstream LLMs. Models are categorized based on their architectural construct: encoder-only, decoder-only, and encoder-decoder. Models stemming from the same lineage or released by identical entities are interconnected with solid lines. Independent research contributions are demarcated by purple lines.

information into the data representation. These models commonly employ training strategies like masked language modeling and bi-directional training. The inception of auto-encoding models can be traced back to 2018 with the release of BERT, a pioneering model that harnessed the encoder-only architecture. This innovation significantly augmented the capabilities of natural language understanding models, enabling them to tackle a diverse range of NLU tasks, including reading comprehension, cloze tasks, and question answering.

1) BERT

BERT [9] stands out as one of the pioneering pre-trained models employing an auto-encoding architecture. Its training strategy primarily hinges on two tasks. The first, Masked Language Modeling (MLM), involves randomly masking tokens in the input data during the preprocessing phase. The majority of these masked tokens are hidden, prompting the model to predict them during training. However, a fraction of these tokens might be substituted with random ones. Occasionally, these replacements are erroneously embedded, compelling

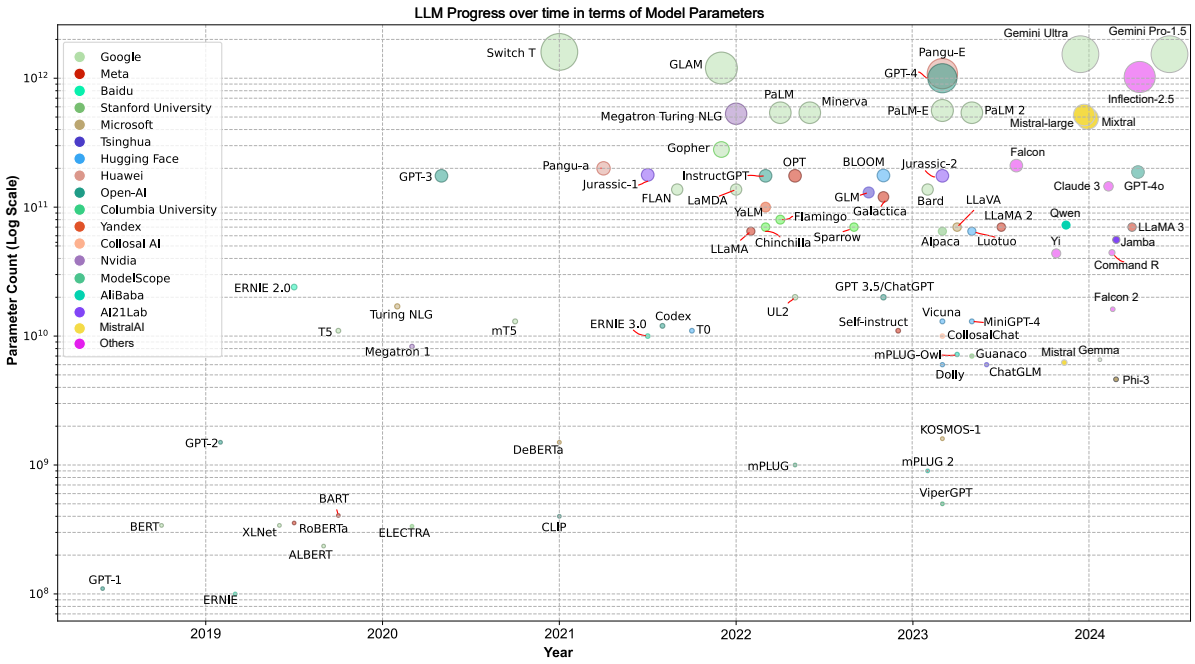

FIGURE 13. **Developments in LLMs and their parameter count reflecting quantitative increase in model sizes across time. The models are color coded based on their research organization.**

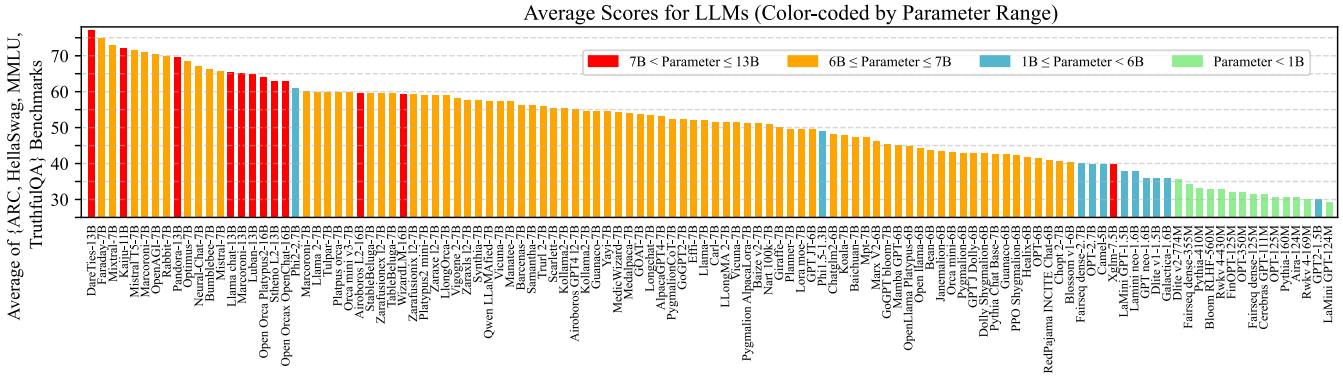

FIGURE 14. **Open-LLM Leader-board benchmark competing various state-of-the-art LLMs across diverse benchmarks, encompassing TruthfulQA, MMLU, ARC, and HellaSwag for a comprehensive evaluation.**

the model to forecast the original tokens using cross-entropy loss. It's worth noting that some of these tokens remain unaltered. This methodology equips BERT with the capability to anticipate contextual information at the token level.

In addition to MLM, BERT introduced the Next Sentence Prediction (NSP) task to capture information at the sentence level. In the NSP task, the model determines whether an input sentence sequentially follows another within the broader context.

For effective NSP task training, both positive and negative samples are used to enhance the model's robustness. BERT also incorporates a unique tokenization system, marking the start of a sentence with [CLS], the end with [SEP], and using the [MASK] token to obscure certain tokens during training. This approach allows BERT to generate contextually accurate and coherent language representations.

2) Varients of BERT

Several BERT variants have emerged to cater to diverse tasks. BERT-wwm [161] employs a whole-word masking (WWM) approach for the MLM task. Contrary to the original BERT's subword-based tokenization, BERT-wwm applies masking to entire words, mitigating issues associated with subwords. BERT-wwm-ext [162] extends BERT-wwm by training on more extensive datasets over increased iterations. SpanBERT [163] offers an expanded MLM version where masked tokens extend to neighboring ones based on a geometric distribution with predefined randomness. SpanBERT omits the NSP task. Efficiency-enhancing adaptations of BERT have emerged. DistillBERT [164] leverages knowledge distillation to derive a streamlined BERT model with half the original's layers. It adopts RoBERTa's [165] optimization techniques, featuring dynamic masking and enlarged batch sizes, while discarding

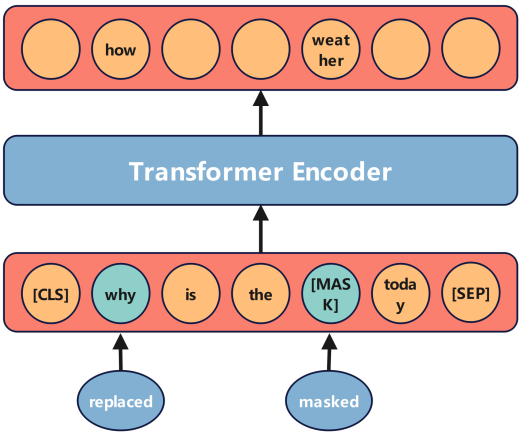

FIGURE 15.
Masked-Language Modeling used by BERT, in this case, the word "how" and "weather" were masked out for BERT to perform prediction tasks.

the NSP task. TinyBERT [166] employs distillation techniques but optimizes BERT's efficiency further. VisualBERT [167] incorporates multimodal support into BERT, pairing it with a convolutional neural network (CNN) to extract features from images. Unlike the original BERT, VisualBERT's token predictions rely on textual context and image-derived information. Lastly, MacBERT [168] introduces synonyms (sourced from word2vec) as MLM replacements and integrates both WWM and n-gram masking. This aims to mirror the objectives of BEiT [169], BEiT v2 [170], and BEiT v3 [171], which fuse a BERT-based encoder with dVAE [172].

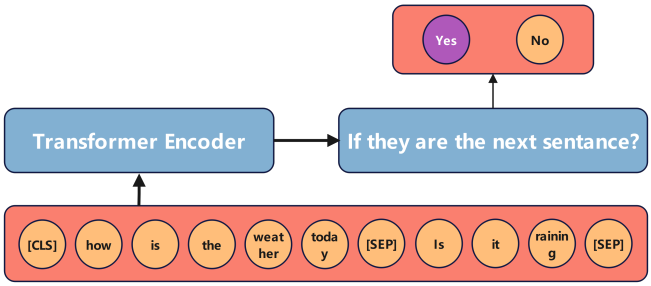

FIGURE 16. Next Sentence
Prediction used by BERT, the model was asked to justify if the sequence "is it raining" should be the next sentence of "how is the weather today"

3) RoBERTa

Several models, while distinct from BERT, owe their genesis to the original BERT framework. One such is RoBERTa [165], designed to enhance the robustness of BERT's training process, primarily through the introduction of a dynamic mask strategy. Contrary to BERT's MLM that employs static masks in a consistent manner for each input sample, RoBERTa's mask selection is dynamic. For any given input sequence, this sequence undergoes multiple masking processes; in their study, the authors adopted a hyper-parameter value of 10. During each masking iteration, a unique set of tokens is selected to be masked, eschewing

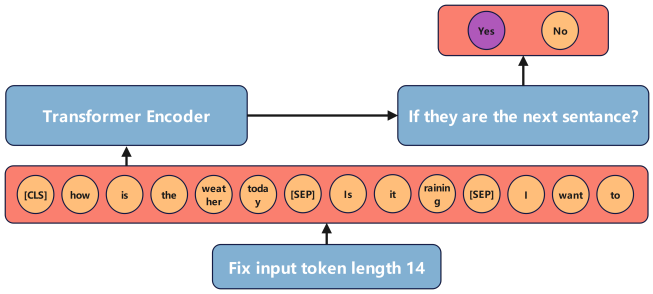

FIGURE 17. changed version of NSP in RoBERTA, different from BERT which used fixed two sentences, RoBERTA would have a fixed length of the whole input sequences and give the NSP input until this fixed length was reached

the repetitive use of static masks. Additionally, RoBERTa's research evaluated various sentence pair configurations for the NSP tasks. Findings suggested that certain sentence-pair configurations adversely impacted the fine-tuning of downstream tasks. Consequently, to maintain document integrity during training, RoBERTa omits the NSP task. To optimize performance, RoBERTa employs larger batch sizes, an expansive training corpus, and deeper training iterations.

A notable variant of RoBERTa, termed RoBERTa-wwm [173], aims to fine-tune RoBERTa for the Chinese corpus. Unique to this adaptation, tokenization and masking occur at the character, rather than word level.

4) ERNIE

ERNIE [8] employs a multi-level masking strategy distinct from BERT to optimize its performance for Chinese languages. This strategy encompasses basic-level masking, which masks out characters or words similar to BERT's MLM approach; phrase-level, where entire phrases are masked rather than individual words; and entity-level, which targets and masks entire entities within the input sequence. Additionally, ERNIE introduces the Dialogue Language Model (DLM) technique. It leverages both genuine dialogues from online forums and artificially generated ones. Within DLM, the model's training loss incorporates its capability to differentiate between authentic and fake dialogues. Like many auto-encoding model variants, ERNIE utilizes an expansive training dataset, which includes content from Chinese Wikipedia and news articles from Baidu. Architecturally, ERNIE is built upon the Transformer-XL framework. Traditional multi-task learning, which trains tasks sequentially, often grapples with efficiency and "forgetting" challenges. To address this, ERNIE 2.0 [174] introduces continual multi-task learning. This method deviates from the conventional sequential approach by integrating new tasks directly into the existing task set, facilitating combined training. ERNIE 2.0 also proposes three types of aware pre-training tasks. The word-aware task is an enhanced version of ERNIE's multi-level masking that incorporates token-documentation relation tasks. It predicts a token's likelihood of appearing in segments and its case (uppercase or lowercase). The structure-aware task involves the permuta-

tion of sentences into sub-sentences, predicting their original order and establishing if sentences are adjacent. Lastly, the semantic-aware task determines relationships between sentences, deducing their cohesive structure and relationships between queries and titles at the information relevance level. ERNIE 3.0 [175] introduces both universal representation and task-specific representation. The former is utilized for natural language understanding tasks, while the latter aids in natural language generation tasks to extract contextual semantic features. A novel knowledge-aware pre-training task is also added, which uses universal knowledge-text prediction to discern the relationships between various knowledge points within the training set. Keeping pace with modern large language models, ERNIE 3.0 expanded its parameters to 10B, supported by a larger dataset. ERNIE has a multimodal variant, ERNIE-VilG [176]. It trains on image datasets through three foundational tasks: object prediction, where associated tokens in the text are masked in relation to image data; attribute prediction, masking attributes of randomly selected objects; and relationship prediction, which targets and predicts the relationship between two objects in an image. ERNIE-Vil 2.0 [177] introduces multi-view contrastive learning. This approach trains image-text pairs based on various pairings, differing from ERNIE-Vil's single image-text pairing strategy, to enhance cross-modality representation.

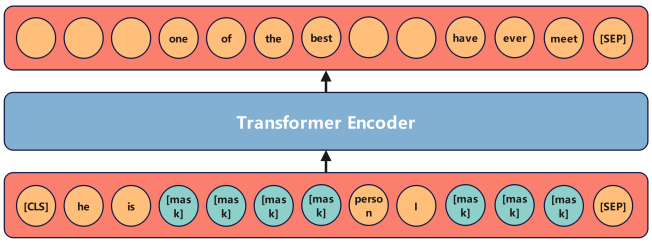

FIGURE 18. ERNIE would mask out the whole related word of the chosen masked word to perform predict, this strategy could improve the capability of the model to infer the relationship between tokens

5) ALBERT

ALBERT [9] was developed to optimize training as model parameters grew. It observed that in earlier models like BERT, XLNet, and RoBERTa, the size of embedding layers was equivalent to that of hidden layers. These embedding layers focused on acquiring context-independent representations, while the hidden layers concentrated on context-dependent representations. Given that the models typically experienced greater enhancements from context-dependent information, it stood to reason that the dimensions of hidden layers should surpass those of embedding layers. To address this, ALBERT introduced the concept of factorized embedding parameterization. Instead of directly transitioning from one-hot encoding to vector embedding, this technique employs one-hot embeddings to first map to a relatively smaller embedding layer. Subsequently, this is mapped to considerably larger hidden layers. This separation ensures that the model doesn't heavily rely on the direct mapping of the vector embedding from the one-hot encoding, which can be restrictive given the differing roles of embeddings and hidden layers. Moreover, ALBERT shared parameters between the feed-forward network and attention layers, leading to further efficiency in training. In a departure from models like RoBERTa which eliminated the NSP task outright, ALBERT evolved the task into Sentence Order Prediction (SOP). This modification expanded upon the NSP concept by transitioning from predicting the subsequent sentence to deducing the order of sentences, addressing the criticism that NSP was rudimentary for the model's potential.

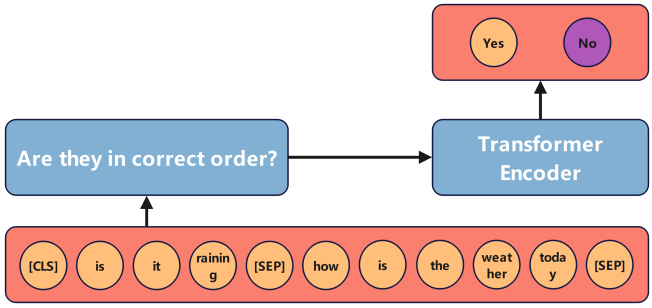

FIGURE 19. ALBERT used sentence-order prediction instead of NSP, during the training, the negative sample were rotated and the model should discriminate if the sentance order was correct

6) ELECTRA

ELECTRA [178] was introduced with the intention of effectively training auto-encoding models on smaller-scale corpora. Unlike traditional Masked Language Models (MLM) that only predict masked-out words, in ELECTRA, the objective is shifted towards predicting all words in the input sentence. Different from the previous models which used different tasks such as masks to enhance the capability of the model during the pre-training process, the key innovation in ELECTRA is to take the feature of generative adversarial network (GAN) [179] which introduced a two-part system: a generator and a discriminator. The generator, akin to the traditional MLM, randomly masks certain tokens and then attempts to predict them. The discriminator's role is to examine each token from the generator's output and determine if it is the actual token from the original input or if it's the token produced by the generator.

By employing this adversarial training technique, the model can more effectively understand contextual information. An important detail is that the token embeddings' parameters are shared between the generator and the discriminator. This shared parameterization not only reduces the model's overall parameters but also ensures consistent representation between the two parts, enabling them to better work in tandem. The result is a model that can be trained efficiently on smaller datasets, yet achieve competitive, if not superior, performance compared to its counterparts trained on much larger corpora.

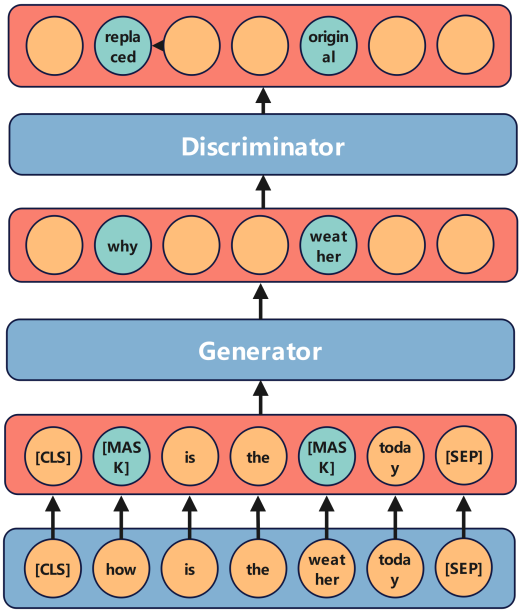

FIGURE 20. ELECTRA used replaced token detection instead of MLM. In this case, the word "how" and "weather" were masked and passed the masked sentence into the generator, the generator produced the predicted masked word "why" and "weather" and used the discriminator to discriminate if the generated word matched the original word, in this case, the word "why" was discriminated as "replaced" and the word "weather" was regarded as the original correct word.

7) DeBERTa

In conventional auto-encoding models, the Masked Language Model (MLM) technique is typically not employed during the fine-tuning stage. This omission often leads to discrepancies between the pre-training accuracy and the fine-tuning accuracy. Furthermore, the attention score is somewhat influenced by the token positions. To address the inconsistency in accuracy and the distribution of token positions, DeBERTa [180] was introduced. This model incorporates a novel attention mechanism termed "disentangled attention," which factors in the relative positional information of tokens when computing the attention score. In addition, DeBERTa employs an enhanced mask decoder that replaces the conventional softmax function with an Extended Mask Decoder (EMD) and adds several transformer layers prior to the original softmax function. Within the Enhanced Mask Decoder, 10% of the relative positional embedding information is substituted with absolute positional embedding, bolstering the attention across diverse positional information.

8) Transformer-XL

Despite of the wide usage of BERT and its derived models, handling long sequences in the standard transformer model presents challenges. Specifically, tokens are segmented into fixed-length sub-sequences. Post-encoding, the model lacks a mechanism to discern relationships between these segments. This limitation hinders the flow of contextual information, as the model cannot ascertain how the fragments relate within the entire sequence, leading to context fragmentation. Transformer-XL [181] is another family of auto-encoding model that addresses this by introducing segment-level recurrence coupled with state reuse. This method integrates a memory layer to retain information from preceding sequences, facilitating the establishment of relationships between sub-sequences during encoding. Such a mechanism not only enables the model to understand semantic connections between segments but also encodes relationships in longer sequences. Moreover, Transformer-XL employs relative positional encoding, replacing absolute positional encoding to mitigate confusion between the current sub-sequence and the entire sequence. XLNet [182] adopts the Transformer-XL architecture to rectify the MLM drawbacks found in BERT. The authors postulated that the introduction of masks might disrupt contextual information construction. To counteract this, XLNet introduces a suite of techniques. One key method is the permutation language model, which factorizes input sequences in varying orders. This replaces BERT's bi-directional training, enabling the model to learn contextual information bidirectionally by alternating factorization permutation orders while sharing parameters. To facilitate factorization permutation, XLNet introduces a two-stream self-attention mechanism that employs two hidden states instead of one, alleviating issues induced by permutation. Additionally, in lieu of complete predictions, XLNet employs partial prediction, akin to BERT's sub-wording, to predict only a token segment, enhancing the model's convergence speed.

B. AUTO-REGRESSIVE MODELS

Auto-regressive models are often referred to as "decoder-only" models. In contrast, auto-encoding models compute attention bi-directionally, enhancing the model's capacity for natural language understanding by perceiving the entire context. While self-attention computes attention scores globally, assessing the correlation between each token in a sequence, auto-regressive models utilize uni-directional attention. This means that the current generation is dependent solely on previously generated sequences; tokens following the current one are masked out. Compared to auto-encoding models, this architecture excels in generation tasks. The uni-directional attention offers superior performance in handling long sequences, which is one of the reasons this architecture is widely adopted in contemporary large language models.

1) GPT

GPT [2] was one of the early auto-regressive models released in 2018, contemporaneously with BERT. It was a pioneer in introducing auto-regression techniques. Before GPT's inception, language model training largely relied on large corpora, often manually or automatically annotated, which were costly to assemble. Many of these earlier models were domain-specific, limiting their zero-shot capabilities. GPT uniquely adopted unsupervised learning, training generatively on unlabeled datasets and subsequently fine-tuning for specific tasks. It incorporated strategies such as natural language inference to assess relationships between sentences, question answering

and commonsense reasoning for semantic comprehension, semantic similarity evaluations, and classification tasks. Despite utilizing a relatively smaller dataset, the BookCorpus [183], it employed a 12-layer transformer encoder.

GPT-2 [184], a successor to GPT, embraced multi-task learning. Building on GPT's foundation, it postulated that with sufficiently large data and model size, supervised tasks could be implicitly learned, as indicated in decaNLP [185]. Consequently, GPT-2 leveraged WebText, a dataset vastly larger than GPT's BookCorpus, comprising over 8 million websites from Reddit, exceeding 40GB. The architecture was also expanded to 48 layers, enlarging the parameter count to nearly 13 times that of GPT.

GPT-3 [186] continued GPT-2's ethos of expanding dataset size and model depth. It adopted an in-context learning training strategy, aiding in faster convergence through the model-agnostic meta-learning method [187]. The distinct fine-tuning phase, present post-pretraining in GPT and GPT-2, was eschewed in GPT-3 due to the reasons mentioned and the enormity of its dataset. GPT-3 sourced data from five diverse repositories, including Common Crawl [188], WebText 2, two iterations of BookCorpus, and Wikipedia, aggregating over 45TB. The model architecture doubled GPT-2's, employing 96 layers of transformer decoder, culminating in a staggering 175B parameters, more than 100 times that of GPT-2.

InstructGPT [189] melded reinforcement learning with human feedback into GPT-3's fine-tuning process. Using the SFT dataset, it established a reward model reflecting human feedback on the model's output through proximal policy optimization, enhancing the model's human-like behavior. Drawing from InstructGPT's technology, OpenAI unveiled ChatGPT [190], a fine-tuned iteration of GPT-3, colloquially termed GPT-3.5.

GPT-4's [191] technical report, while not fully revealing its strategies, highlights some salient features. GPT-4 can handle both image and text inputs, producing textual outputs, classifying it as a genuine MLLM. This flexibility broadens GPT-4's task repertoire, especially for multimodal requirements. While ChatGPT, based on GPT-3.5, supports up to 4096 tokens (roughly 3000 English words), GPT-4 manages a remarkable 32767 tokens, or about 25000 words, enabling the processing of more extended text sequences with heightened accuracy. GPT-4 emphasizes security, addressing challenges like Adversarial Usage, Unwanted Content, and Privacy Concerns through strategies like RLHF (reinforcement learning with human feedback), real-world use case simulations, and an adversarial testing program. Additionally, GPT-4 allows users to use precise prompts, granting more control over the model's behavior.

Several models have been derived from or inspired by the GPT series. GPT-Neo [192] seeks to offer an open-sourced version of GPT-3 for local deployment. GPT-GNN [193] integrates pre-training on graph neural networks to understand node interrelations. GPT-J [194] is GPT-inspired, grounded on the Mesh-transformer-JAX [195]. GPT-NeoX [196] succeeds GPT-Neo, integrating parallel computing from GPT-J using the Deepspeed [197] framework. DialoGPT [198], an extension of GPT-2, aims to refine text generation using maximum mutual information to elevate hypothesis ranking.

2) Pathways and PaLM

The "Pathways" architecture, as introduced by Google [199], represents a significant shift in AI design. Instead of the traditional method of constructing distinct models for individual tasks, Pathways proposes a single AI system capable of generalizing over thousands, if not millions, of tasks. This versatility is attributed to the "mixture-of-experts" (MoE) concept. By constructing a single model that can be trained on extensive datasets encompassing both text and code, different tasks or inputs are managed using a gate function, directed by the respective experts. A notable feature of models built using the Pathways architecture is their ability to be fine-tuned for specific tasks through few-shot learning. This approach empowers the model to master a new task using only a handful of examples, ensuring efficiency, versatility, and precision in its applications. PaLM [200] is the first language model trained by Pathways architecture which is a model contains up to 540B number of parameters due to the sparsity architecture of Pathways. PaLM used a variant version of traditional transformer decoder which made the modification below:

1) used SwiGLU [201] activation function
2) used multi-query attention, sharing all parameters between all heads of multi-head attention to boost the efficiency of decoding
3) put the feed forward network in parallel with attention layers
4) used RoPE [202] positional embedding
5) Remove all bias in the neural network
6) sharing the embedding among the input and output

The "Pathways" architecture's utilization of few-shot learning, as detailed in PaLM, yielded state-of-the-art outcomes closely paralleling human performance across four tasks. Moreover, the introduction of the BIG-bench, encompassing 150 tasks, further highlighted its prowess, especially in comparison to models like GPT-3.

The Embodied Pathways Language Model, or PaLM-E [203], is an evolution of the preceding PaLM. It distinguishes itself as the premier Large Language Model (LLM) developed using Google's Pathways architecture. This architecture was crafted to facilitate sparse activations across multimodal and multi-task scenarios, implying that for a distinct task, only a section of the model activates, curtailing computational intricacy. PaLM-E, much like Microsoft's KOSMOS-1, is adept at deciphering both natural language and imagery, and boasts potential applications in robotics for accomplishing embodied tasks. Training for this model is conducted on the expansive GEM dataset, encompassing both textual and visual data sources like ImageNet [204], COCO [205], and Visual Genome [206]. PaLM-E integrates two primary components: the language pathway, rooted in the Transformer-XL framework, and the visual pathway, modeled after the

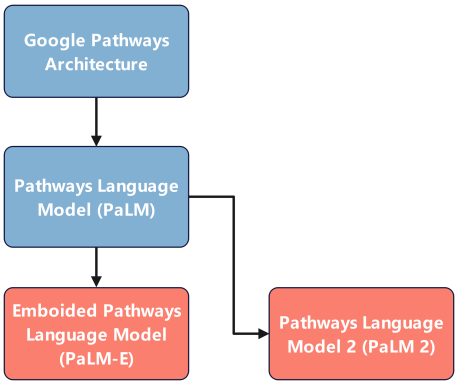

FIGURE 22. The relationship between models under PaLM family by Google

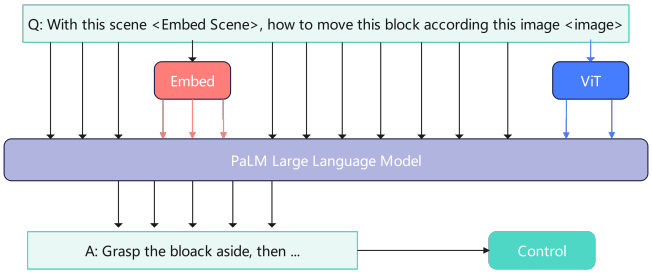

FIGURE 23.
Architecture of PaLM-E, first the visual information would go pass the ViT, and the scene would be embedded into vector format, then all the token embedding would go through a PaLM model to generate the response text, that response text would then be transferred to the control signal

Vision Transformer (ViT) framework. A fusion module bridges these pathways, enabling seamless integration of both modalities. Its efficacy is demonstrated through exceptional performances across a gamut of tasks such as image captioning, visual question answering, and embodied navigation. Architecturally, PaLM-E synergizes two distinct models: the PaLM, which serves as a textual decoder, and the ViT 22B [207], a dedicated visual transformer. The PaLM delivers linguistic processing proficiency, while the ViT 22B specializes in image processing. The default model amalgamates a 540B PaLM model and a 22B ViT model, cumulating to a massive 562B parameters. In terms of training data, PaLM-E benefits from a staggering 780B tokens, derived from diverse sources like social media, web content, Wikipedia, and GitHub. Meanwhile, the ViT 22B module trains on the JFT dataset [208], encompassing roughly 4B semi-automatically annotated images. The model's input representation strategy mirrors KOSMOS-1 [209], where visual or robotic-related inputs receive specific tags, and distinct encoding techniques interpret the content within these tags. Directly, textual data is channeled into the language model. Additionally, the Minerva model [210], fine-tuned from a vast 118G dataset teeming with scientific publications and mathematical expressions, is predicated on PaLM and targets challenges in scientific and mathematical domains. Lastly, PaLi [211], another PaLM derivative, tackles text-vision quandaries using an image-and-text to text-only paradigm, incorporating models like mT5-XXL [212], ViT-G [213], and ViT-e [214].

The PaLM-2 model [215] augments the capabilities of its predecessor, the PaLM, with enhanced performance in mathematics, coding, inference, and multi-language tasks. Leveraging JAX and TPU technology, the primary objective of PaLM-2 is to amplify the efficacy of the PaLM model while simultaneously utilizing fewer parameters. In contrast to PaLM-E, PaLM-2 does not offer multi-modal support. The PaLM-2 architecture encompasses four variants differentiated by parameter size: Gecko, Otter, Bison, and Unicorn. These versions ensure versatility, catering to varied deployment needs across diverse platforms. Notably, the Gecko model, being the most compact, is optimized for deployment on mobile devices. In the realm of medical question-answering (QA), the Med-PaLM 2 [216] was introduced. This specialized version, derived from PaLM-2, employs fine-tuning techniques centered on the ensemble refinement approach applied to medical datasets. Impressively, it registered an accuracy rate of 86.5% on the MedQA benchmark [217], marking a significant improvement from its antecedent, the Med-PaLM [218].

3) Microsoft KOSMOS-1

The KOSMOS-1 model [209] is underpinned by the magneto transformer [219], a derivative of the original transformer architecture. The primary distinction between the two lies in the magneto transformer's integration of a layer normalization between its linear layer and the activation function, a modification that enhances training stability and scalability. Specifically designed with 24 decoder layers, each having 2048 dimensions, 8192 hidden size, and 32 attention heads, KOSMOS-1 boasts approximately 1.6 billion parameters.

Rather than directly processing images, KOSMOS-1 expedites training by harnessing the CLIP ViT-L/14 model to capture image features with 1,024 dimensions. Moreover, it employs the XPOS technique, leveraging length-extrapolation to reconcile disparities in length between training tokens and predicted tokens. These innovations equip KOSMOS-1 with the versatility to excel in a range of tasks, such as image captioning and visual question answering, underscoring its prowess in multimodal learning.

KOSMOS-1 is enriched by three primary datasets. The text corpus includes The Pile, a vast English text dataset tailored for LLMs, and other resources like Common Crawl, CC-Stories, and RealNews. Image-caption pairings are sourced from datasets such as English LAION-2B, LAION-400M, COYO-700M, and Conceptual Captions, all of which were acquired through web crawling. Additionally, there's an interleaved image-text compilation featuring combined image and text fragments. This data is culled from an initial collection of 2 billion web pages, which was subsequently condensed to 71 million pages in the finalized dataset.

For data preprocessing, text sequences were designated with the <s> tag, while images received the <image> label. Notably, while KOSMOS-1 also extends support to audio

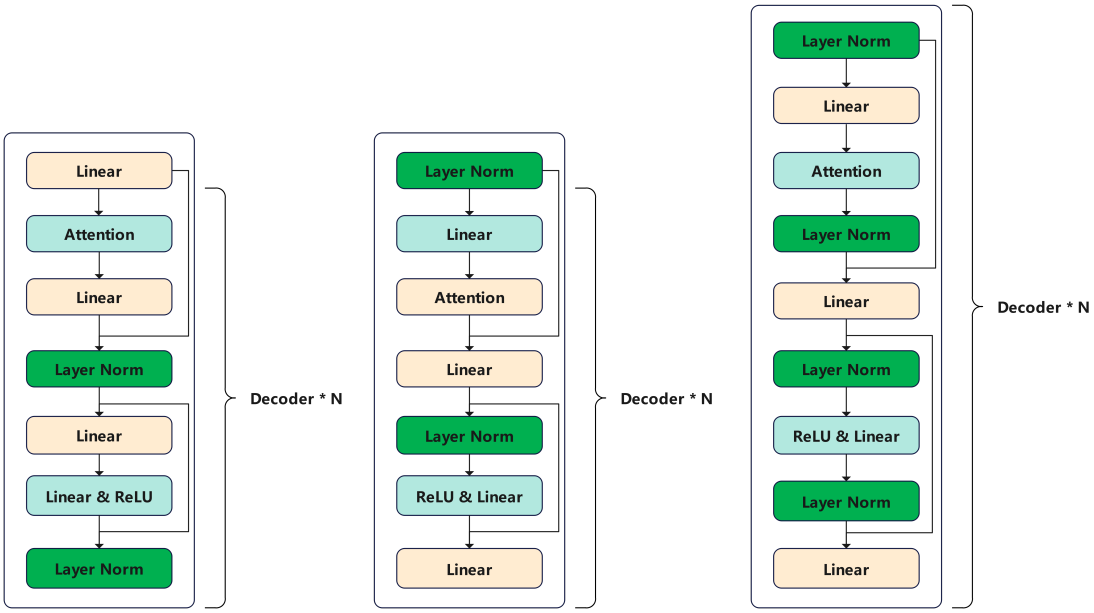

FIGURE 24. Structure of the foundation transformer used in KOSMOS-1

sequences, the associated paper lacks comprehensive details on this aspect, hinting at its potential developmental stage.

Conclusively, KOSMOS-1 signifies a monumental stride in the evolution of multimodal large language models, poised to have transformative impacts on both natural language processing and computer vision research arenas.

4) Megatron

Nvidia Megatron [220] is a framework proposed by Nvidia to solve the parallel computing of the LLM training, it mainly used two strategies to boost the model training and relief the shortage of VRAM during the training process. Inter-layer parallel, also known as tensor parallel, aimed to divide a single model into several layers and deploy each layer into separate devices, while intra-layer parallel, also known as tensor parallel, divided a single model into multiple layers to deploy on different devices. The final output would be concatenated from these model segments after processing directly. Data parallel, divides the whole dataset into different devices while each device still holds a full copy of the model to boost the training process of the model. Megatron LM [221] is a LLM trained with Megatron framework with 8.3B parameters and trained distributively on multi-GPU with both tensor parallel and data parallel strategies. While its successor, Megatron LM 2 [222], introduced a strategy called PTD-P which combined all of the tensor parallel, pipeline parallel and data parallel to train the model which it claimed, is able to train a model over 1T size with IO 502 PTFLOG/S. Megatron LM 3 [223] introduced sequence parallel which divide the non-tensor part of the model along the sequence dimensions and also introduced a reduce-scatter operation which reduce the VRAM required for the activation function. Turing NLG [224] was an early LLM released by Microsoft which aimed to solve the similar parallel computing problem as Megatron models, which used DeepSpeed [197] and ZeRO [225] to boost the training based on distributed computing which is a breakthrough of both hardware and software with reduced training time around $2/3$. Turing NLG contains 78 layers of transformer decoder and 170B parameter which is the largest model when it was released following by Megatron LM. ChatGPT Nvidia's Megatron [220] is a framework specifically designed to address the challenges associated with the parallel computing of large language model (LLM) training. The architecture primarily incorporates two strategies to expedite model training and alleviate the constraints of VRAM during the training phase. Firstly, the intra-layer parallelism, also known as tensor parallelism, partitions the layers of a single model into fragments that are then distributed across different devices. Once these individual segments process the data, their outputs are concatenated to produce the final result. In contrast, inter-layer parallelism involves segmenting a model into distinct layers, with each layer being allocated to a separate device. Another strategy, data parallelism, involves distributing the dataset across multiple devices, but each device retains a full copy of the model, thereby accelerating the model's training process.

Megatron LM [221], a large language model nurtured using the Megatron framework, boasts 8.3B parameters. The model is trained distributively across multiple GPUs, leveraging both tensor parallelism and data parallelism. Its successor, Megatron LM 2 [222], integrated a method termed PTD-P, which synergistically combines tensor parallelism, pipeline parallelism, and data parallelism. This integrated approach empowers the framework to train models exceeding 1T in size, achieving an impressive IO of 502 PTFLOG/S. Subsequently, Megatron LM 3 [223] incorporated sequence

TABLE 6. Training dataset of Megatron-Turing NLG

Dataset	Tokens (B)	Weights (%)	Epochs
Books3	25.7	14.3	1.5
OpenWebText2	14.8	19.3	3.6
Stack Exchange	11.6	5.7	1.4
PubMed Abstracts	4.4	2.9	1.8
Wikipedia	4.2	4.8	3.2
Gutenberg (PG-19)	2.7	0.9	0.9
BookCorpus2	1.5	1.0	1.8
NIH ExPorter	0.3	0.2	1.8
Pile-CC	49.8	9.4	0.5
ArXiv	20.8	1.4	0.2
GitHub	24.3	1.6	0.2
CC-2020-50	68.7	13.0	0.5
CC-2021-04	82.6	15.7	0.5
RealNews	21.9	9.0	1.1
CC-Stories	5.3	0.9	0.5

parallelism, which segments the non-tensor components of the model along the sequence dimensions. This edition also introduced a "reduce-scatter" operation, significantly diminishing the VRAM demands of the activation function.

On a parallel trajectory, Microsoft's Turing NLG [224] emerged as an early contender in the LLM space, targeting parallel computing challenges akin to those addressed by the Megatron models. Turing NLG integrates the capabilities of DeepSpeed [197] and ZeRO [225], pioneering advancements in both hardware and software realms. This integration has culminated in a substantial reduction in training time, approximately by two-thirds. Architecturally, Turing NLG comprises 78 transformer decoder layers and 170B parameters. At the time of its release, it held the title of the largest model in its category, subsequently succeeded by Megatron LM.

Megatron-Tuiring NLG [226] is the successor of Turing NLG in combination with the Megatron architecture which contains over 530B of the parameters over the Turing NLG. It was trained based on 280 Nvidia A100 GPUs and contains 105 layers of transformer deocder. The training data was from a variety of sources shown below:

There are other models that leverage the power of parallel computing from Megatron architecture, BioMegatron [227] used Megatron to train a LLM based with biomedical domain specific and Megatron-CNTRL [228] which proposed a framework of LLM that has controllability output by keyword prediction, knowledge retrieval, contextual knowledge rank and conditional text generation.

5) LLaMA

The LLaMA model was introduced by Meta in 2022 and made open-source. Its primary aim was to enhance model capabilities while maintaining a smaller size suitable for local deployment [10]. LLaMA employed three distinct strategies. Firstly, it adopted the RMS Pre-Norm [229] in the transformer decoder, replacing the conventional layer normalization. This modification eliminated the re-centering operation, retaining only the re-scaling operation, thereby facilitating smoother gradient descent. Additionally, LLaMA utilized methods similar to PaLM, incorporating both SwiGLU [201] and RoPE positional embedding. The LLaMA model suite consists of four variants: 7B, 13B, 33B, and 65B parameters. Despite having significantly fewer parameters than GPT-3, Meta suggested that LLaMA could be locally deployed. However, the performance of LLaMA, with its reduced parameter count, did not significantly surpass that of GPT-3.

Following LLaMA, a series of derivative models emerged. Alpaca [230] sought to fine-tune the original LLaMA model using over 52,000 fine-tuning data samples extracted from the text-davinci-003 model [231] developed by OpenAI for GPT-3. The resulting Alpaca model achieved performance comparable to text-davinci-003 but at a reduced cost. Subsequently, Guanaco [232] was introduced as Alpaca's successor, integrating block-wise k-bit quantization. This feature marked the LLaMA series' first foray into quantization methods, compressing the model by converting the original FP32 data format to a more compact int8. Additionally, Guanaco employed low-rank adapters (LoRA) to keep model parameters constant, only adjusting the optimizer with a smaller dataset batch. This approach further reduced the fine-tuning costs. Alpaca-LoRA [233] combined Alpaca with the LoRA technology. Vicuna [234], with its 13B parameters, represented a fine-tuned LLaMA model derived from dialogue datasets from ShareGPT [235], achieving approximately 90% of ChatGPT's performance.

Distinct from earlier iterations, which primarily fine-tuned the original LLaMA, Dolly [236] employed GPT-J-6B [194] to emphasize the importance of fine-tuning over baseline models. Dolly v2 [237] transitioned to using the databricks-dolly-15k dataset and replaced GPT-J-6B with Pythia [238], making the model more accessible for business applications. Koala [239], applied to FastChat [240], adopted approaches similar to Vicuna, drawing from dialogue data for its fine-tuning. LLaMA 2 [10], a direct successor of LLaMA, presented three versions with 7B, 13B, and 70B parameters. It increased the original LLaMA training dataset by approximately 40%. LLaMA 2 introduced the grouped-query attention (GQA) strategy, which grouped attention calculations between K and V values into sets of 8, thereby optimizing attention score computations. A notable distinction between LLaMA 2 and its predecessor was the doubling of the context length, coupled with enhanced data cleaning methods. Baize [241] offered a novel pipeline leveraging ChatGPT to autonomously generate new fine-tuning data.

Several LLaMA derivatives aimed to achieve multi-lingual capabilities. Chinese-Vicuna and Luotuo (also termed Chinese-alpaca-lora [242]) fine-tuned the LLaMA model using the LoRA approach to support Chinese, among other languages. Chinese Llama Alpaca 2 [243] expanded upon this, incorporating datasets from various languages, including German and French.

Others sought to integrate multimodal support into LLaMA. LLaMA adapter [244] introduced a pipeline adapting LLaMA for visual instruction-following, yielding a smaller and quicker fine-tuning model. Its successor, LLaMA

adapter V2 [245], preserved instructions from its predecessor while refining the image-text projection to enhance visual-text alignment. MiniGPT-4 [246], grounded in the Vicuna model, implemented a two-phase pre-training approach. After freezing both the language model and visual encoder, it constructed a new image-text dataset for fine-tuning MiniGPT-4, ensuring superior visual-text alignment. Both Chinese-LLaVA [247] and LLaSM [248] were derived from LLaMA 2 but incorporated multi-modal support. Visual-LLaMA [249] utilized a technique consistent with KOSMOS-1 and PaLM-E, merging visual and text tokens for training. Video-LLaMA [250] integrated both audio and visual information, using a two-layer Q-former [251] for video embedding. This model underwent training on the Webvid-2M dataset and the image captioning dataset from LLaVA [158], a visual-language model built on 150K multimodal data samples generated by GPT-4. Post pre-training, fine-tuning instructions from MiniGPT-4, LLaVA, and VideoChat [252] were applied. The audio data in Video-LLaMA was processed similarly, but with the inclusion of the ImageBind-Huge encoder [253] for audio information embedding.

6) Gopher and DeepMind

Gopher, introduced by DeepMind, is a large language model (LLM) with a training pipeline that encompasses six models, varying in parameter count from 44M to 280B [254]. This model employs the RMS Pre-norm strategy [229] and incorporates relative positional embeddings using a 32,000 token SentencePiece tokenizer [255]. For training and evaluation, the Gopher utilizes the JAX [256] and Haiku [257] frameworks. Parallel computing is facilitated by JAX pump, and the model is trained using the MassiveText dataset.

Chinchilla, also a product of DeepMind and the successor to Gopher, posits that as the size of the model increases, the number of tokens trained should proportionally increase [258]. It suggests that the optimal Gopher model should be four times smaller when trained on a dataset that is four times larger. Various fine-tuning strategies based on Gopher were explored to ascertain the optimal ratio between model size and training data. These included altering the number of tokens per batch, maintaining consistent FLOPs, and training with a parameterized loss function. The latter was identified as the optimal approach.

DeepMind further developed a visual model named Flamingo, with 80B parameters, tailored for few-shot learning [259]. It employs the Perceiver resampler, training with a pre-established visual model known as NFNet [260]. During the pre-training phase of the language model, NFNet remains static. Subsequently, the Perceiver resampler is integrated with the frozen language model and the visual model, yielding visual representations. The language model is then fine-tuned, leveraging these visual representations, which strengthens the nexus between the language and visual models, resulting in state-of-the-art performance.

7) Other auto-regressive models

Jurassic-1, introduced by AI21 Lab in collaboration with the AI21 Studio, was developed with the objective of offering an open conversational API [261]. At the time of its release, the Jumbo edition of Jurassic-1, with its 178B parameters, was heralded as the most intricate and expansive model available [262]. This model was trained on a corpus encompassing 250K labeled datasets. AI21 asserted that the dataset underpinning the Jurassic-1 model was quintuple the size of concurrent datasets. Succeeding Jurassic-1, Jurassic-X integrated the Modular Reasoning, Knowledge, and Language (MRKL) system—a composite of mixed expert data extraction from multiple databases. The outputs from these extractions are then processed by the language model, achieving a balance between universality and sparsity in large language models [263]. In 2023, AI21 Lab unveiled the Jurassic-2 model [264] with enhancements spanning multilingual capabilities, accuracy, and latency.

Anthropic launched the Claude model series, comprising two iterations: Claude [265] and Claude 2 [266]. The foundational ethos of Claude echoes the principles laid out in the InstructGPT paper—namely, the creation of AI language models that are helpful, honest, and harmless [189]. With this framework, Claude was trained on a dataset meticulously curated to align with these objectives, blending datasets that were both assistance-focused and harm-avoidant. This amalgamated dataset bolstered the performance model's propensity for helpfulness while eschewing potentially detrimental instructions. Claude 2, the subsequent model, showcased amplified performance metrics, supporting extended sequences, enhanced coding proficiency, and heightened security measures. Anthropic posited that Claude 2 boasts a security standard twice as robust as its predecessor, Claude 1.3.

Falcon, a pioneering model accessible to the public, stands as a worthy counterpart to several proprietary models [267]. It comprises three variants: Falcon-7B, Falcon-40B, and Falcon-180B. Predominantly trained on the RefinedWeb dataset—a refined iteration of the CommonCrawl dataset [188], [268]—Falcon harnesses multi-query attention, mirroring the PaLM model. This approach substantially curtails the memory overhead of the K, V values during training, slashing memory usage by factors ranging from 10 to 100.

OpenAI, beyond the GPT series, has released several domain-specific models. DALL-E [269] is a model designed for image generation. It incorporates three distinct models: the dVAE [172], which encodes and compresses the input image; a BPE Encoder combined with a transformer for auto-regressive training; and CLIP [270] to measure the similarity between text-image pairs. DALL-E 2 [271] employs CLIP to align the feature spaces of both text and image data. Subsequently, it introduces a module named "prior" that uses caption data to embed the data from the CLIP model, then decodes this data into an image. Notably, DALL-E 2 has approximately 3.5B parameters, substantially fewer than DALL-E's 12B parameters. DALL-E 3 [272], released in 2023, is reportedly built on ChatGPT, though detailed technical

information had not been disclosed at the time of this survey. Whisper [273] is designed for audio-to-text conversion and boasts 1550M parameters. The training of Whisper utilized over 680K audio samples from 98 languages, supporting multi-tasking in a supervised learning context. Each audio sample was segmented into 30-second chunks and processed into log-Mel-frequency cepstrum. Codex [274] targets coding challenges, comprising 12B parameters with [275] which is also a model that is specific on coding tasks. It is trained using a dataset amalgamated from GPT data and open-source code from more than 5400 GitHub repositories, which, post-cleaning, amounted to 159G. Its efficacy was validated using the HumanEval dataset, which includes 164 manually generated programming problems to ensure the output's accuracy.

In addition to the PaLM series model developed using the Pathways architecture, Google has unveiled several domain-specific models. Meena targets end-to-end question answering and employs the evolved transformer architecture [276]. Its performance is assessed through human evaluation and perplexity metrics. LaMDA [277] employs knowledge-based queries, building on Meena to endow the model with the ability to answer industry-related knowledge questions. ALIGN [278] is designed to tackle representation learning for visual-textual data. It processes an extensive dataset containing over 1B noisy image captions, incorporating EfficientNet [279] for image-text matching, retrieval, and visual classification tasks. The objective of GaLM [280] is to harness the sparsity of LLM for efficient few-shot learning. The model is trained on a colossal dataset featuring 1600B tokens, predominantly sourced from websites. GaLM utilizes the MoE architecture for training and houses approximately 1.2T parameters across 64 experts. Intriguingly, during inference, only 97B of these parameters are activated, enabling rapid response times. Lastly, Gemini [281] is a model that recently released by Google as their next-generation of large language model, which contains Gemini Ultra, Gemini Pro and Gemini Nano, Genmini.

In June 2023, Microsoft proposed Phi-1, a model with just 1.3 billion parameters. Its goal is to achieve high accuracy performance with a small model, highlighting the significance of high-quality data during the pre-training phase. They stated that they used exercises using 1B tokens and GPT-3.5 along with 6B tokens from online and artificially generated textbooks, all of which were of "textbook quality". Following the release of Phi-1, Microsoft also released Phi-1.5 [282], which was the same size as Phi-1, and Phi-2 [283], which was their most recent model and a foundation model with 2.7B of parameters to exhibit the potential capability on small language models (SLMs).

mPLUG is a series of models developed by Alibaba, tailored for multimodal support in Large Language Models (LLMs). The conceptual foundation of mPLUG [284] draws inspiration from the modularity of the human brain. In this architecture, each modality is treated as a separate module, allowing for specialized task handling. Conversely, mPLUG-2 [285] adopts a unified model approach, managing all tasks within a single model, yet through different modules. mPLUG-Owl is a conversational LLM that has been open-sourced. It employs a Vision Transformer (ViT) to learn from visual-language captioning data. Subsequently, the model is fine-tuned using LoRA to align both uni-modal and multi-modal data, while preserving the foundational visual-text modules trained in the initial phase. There are other models like MM-REACT [286] and HuggingGPT [287] that serve to facilitate collaboration across various modalities. Additionally, models such as Youku-mPLUG [288], mPLUG-DOCOWL [289], and MultiVENT [290] are derivatives of the original mPLUG design.

Numerous institutions, companies, and research affiliations have released auto-regressive models to push the boundaries of machine learning. AlexaTM [291], a multi-lingual model housing 20B parameters, has achieved state-of-the-art performance, especially for low-resource languages. PLATO [292] leverages discrete latent variables to encapsulate invisible background knowledge. Its successor, PLATO-2 [293], refines the original by expanding both the training data and the number of parameters, incorporating a curriculum learning approach. The WuDao series boasts models like WuDao 2.0 [294], one of the most extensive LLMs with 1.75T parameters, and WenLan [295], a 5.3B parameter model tailored for Chinese-English visual captioning, rooted in a 650M image-captioning dataset and utilizing the Deep Structured Semantic Model (DSSM) [296] technology. Cogview [297], a 4T parameter Chinese multimodal LLM, and Lawformer [298], an early LLM dedicated to the legal domain, are noteworthy contributions. OPT [299], developed by Meta, is an endeavor towards open-sourcing LLMs, with models ranging from 120M to 175B parameters. Its enhanced version, OPT-IML [300], comprises two models with 30B and 175B parameters, respectively, and is fine-tuned using datasets spanning over 2000 languages. YaLM [301], a 100B LLM by Yandex, is trained on 1.7T text data sourced from websites, books, and other mediums. BLOOM [302] is a comprehensive model with 176B parameters, trained on data from 46 natural languages and 13 programming languages, representing the collaborative efforts of over 1000 scholars. Lastly, Galactica [303], developed in partnership between Meta and Papers with Code, mirrors the ambitions of GLaM in addressing scientific challenges. Mistral [304] claimed it uses mix-of-expert outperformed GPT-3.5 with smaller model size with the usage of mix-of-expert.

C. SEQUENCE TO SEQUENCE MODELS

Another prominent architecture in the domain of large language models combines the features of both Auto-encoding and Auto-regressive models, known as the sequence-to-sequence model. Typically, it employs the complete Transformer framework, encompassing both encoder and decoder structures. This hybrid model amalgamates the strengths of its predecessors, inheriting the natural language understanding capabilities from the encoder and the generation competencies from the decoder. To integrate the functionalities of both

TABLE 7. Training Strategies used in BART

Token Masking	Replace tokens with masks randomly
Token Deletion	Delete tokens randomly
Text infilling	Replace span with difference length with masks
Sentence Permutation	Permute the order of several sentences
Document Rotation	Rotate the order of the sequences inside a document with a randomly chosen token

encoder and decoder, cross-attention is implemented between their respective layers, facilitating the interplay between different sequences. Owing to their distinctive advantages over models that solely use an encoder or decoder, sequence-to-sequence models are predominantly chosen for conditional generation tasks like summarization and machine translation. Compared to the previous two architectures, which multiple models may share one single origin, sequence to sequence models are more independent, which means it has fewer models derived from previous predecessors but normally, they formed their own family by them own regarding different tasks hence cause less consistency compared with the auto-encoding models and auto-regressive models.

1) BART

BART [305] represents a significant development in the Sequence-to-Sequence model lineage and stands as one of the earliest models harnessing the full Transformer architecture. It masterfully combines BERT's [3] bi-directional encoder characteristics with the auto-regressive decoder features from GPT. In training BART for sequence generation tasks, the Google research team employed a reconstruction loss, defined as the cross-entropy between the conditionally generated output based on the input and the actual ground truth. They also enhanced the masking technique inherited from BERT. Several masking strategies were adopted:

BART's training strategies can vary depending on the targeted downstream tasks. For sequence classification tasks, both the encoder and decoder process the sequence, and the final state of the output is utilized. Similarly, for token classification tasks, the decoder's final state serves as the definitive representation for each token, though these tokens are considered independent. In machine translation tasks, the encoder's embedding is supplanted with an additional encoder termed the "randomly initialized encoder," which can leverage a disjoint vocabulary.

In a step forward, mBART [306] harnesses corpora from multiple languages, bestowing BART with multi-linguistic capabilities. This version was fine-tuned from the foundational BART model.

2) Based on T5

T5 [11] introduced a universal framework in the realm of pre-trained models. One of its major contributions was presenting a clear guide for future researchers concerning parameter and architecture selection. The model underwent training on 750G of tokens, encompassing 11B parameters.

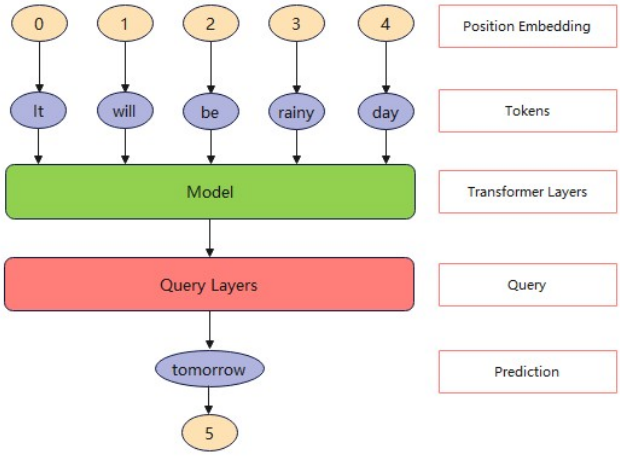

FIGURE 26. The architecture of Pangu-˚, the query layer proposed in that model was used after the embedding passed through the transformer layers, then the prediction of the next token wouold be produced from the query layer with position embedding

In T5's approach, every NLP task was conceptualized as a text-to-text task, leading to the adoption of the Sequence-to-Sequence architecture. The model utilized three distinct masking techniques: fully-visible masks where attention scores are computed from all input and output tokens; causal masks, similar to GPT's, where scores are derived only from the current token and its antecedents, predicting the subsequent token based on previously generated content; and causal masks with prefix, where the input sequence employs a fully-visible mask, but for the output's predicted tokens, only causal masks are used.

The study also evaluated three architectures: the encoder-decoder, the encoder-only (as seen in GPT-2), and the Prefix LM. In the latter, while the encoder can perceive bi-directional information, the decoder is limited to previously generated tokens. Furthermore, the C4 dataset, crafted from the Common Crawl database [188], emerged from this work. The dataset involved a refinement of about 750GB of web data, retaining lines that ended with standard punctuation, eliminating offensive words, omitting lines containing Javascript, discarding pages that resembled code, excluding lorem ipsum passages, and ensuring sentences that appeared more than three times were uniquely preserved.

1) Only Kept lines end with normal punctuation
2) Deleted all bad words
3) Removed lines with Javascript
4) Deleted all pages like code
5) Removed lorem ipsum
6) Only kept once for same sentences appears more than three times

Building on T5's foundation, mT5 [212] aimed at machine translation tasks within a multilingual context. It borrowed strategies from the original T5 model but incorporated more vocabulary to ensure a broader linguistic coverage. This was realized by drawing samples from languages with

limited training data, using an approximation technique as suggested by [307]. T0 [308], evolving from T5, explored the enhancement possibilities of LLM focusing on prompt engineering for multitasking and robustness. Trained atop T5-LM, T0 incorporated training prompts from 177 datasets, amassing 2073 prompts in total. The performance indicated marked improvements in comparison to GPT-3. In a departure from FLAN [307], which also built upon T5, T0 retained the encoder-decoder architecture, whereas FLAN was solely anchored on T5's decoder.

3) Pangu

Pangu is a series of models comprising *Pangu* $-\alpha$ [4], *Pangu*$-$*Coder* [309], and *Pangu*$-\Sigma$ [5]. Unlike its precursor, *Pangu* $-\alpha$, the *Pangu* $-$ *Coder* was specifically designed as a decoder-only model tailored for generation tasks in both Chinese corpus and code generation. In contrast, *Pangu* $-\Sigma$ adopts an Encoder-Decoder architecture. Introduced in March 2023, the primary goal of *Pangu* $-\Sigma$ was to develop a large language model focusing on the Chinese corpus, while also being adept at multilingual tasks. Huawei announced that this model was the first China-centric LLM boasting a colossal 1T parameters size, approximately 1.085T, and underwent training with 2.17T data spanning over 300B tokens. The data sources for its training included 200G from WuDaoCopora 2.0 [310], 100G from CLUECorpus 2020 [311], 800G from the Pile [312], and 750G from C4 [11]. Given its capability to execute code-related tasks, the *Pangu* $-\Sigma$ also incorporated 157G of Python code and 161G of Java code.

Two innovative technologies bolstered the model's efficiency. The first was a novel sparsity technique termed "Random Routed Experts" (RRE). Implemented in two phases, the initial step involved grouping the experts based on their parameters by the same task. Then, for every prompt, RRE deviated from the conventional MoE approach. Instead of allocating tokens to a specific expert with the most compatible matchup governed by a gate function, RRE would haphazardly assign an expert for a token. This gate-free procedure enabled researchers to derive sub-models from *Pangu* $-\Sigma$, facilitating their application to other downstream activities like translation and chatting. The second technological breakthrough was the "Expert Computation and Storage Separation". This mechanism strategically distributed model training across clusters, resulting in a remarkable 69,905 tokens-per-second I/O efficiency and a substantial reduction in communication overhead between servers and devices.

4) Switch Transformer

Switch Transformer [313] employs the sparsity inherent in LLMs to accelerate both training and inference. The term "switch" in its name refers to the shifting between experts. Thus, its primary technology is the Mixture of Experts (MoE). In the architecture of this model, there are several experts, each having distinct parameters that are regulated by a gating function. For any given input or prompt, only certain sections of the model are activated based on the input parameters,

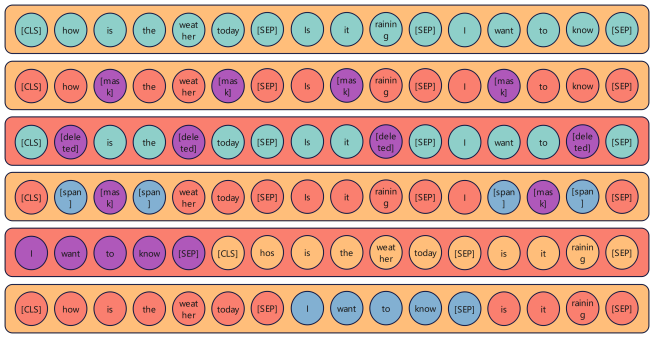

FIGURE 27. Noise used by BART. 1. Original sequences, different sequence denoted by different colour 2. Token masking, in this case, the word "is, today, it, want" were masked 3. Token deletion, the word "how, weather, raining, know" were deleted 4. Token infilling, token "is, to" were masked and it was required to infill the token "how, the" and "want, know" spanned from the masked tokens 4. Document rotation, in this case the last sentence "I want to know" was rotated to the front of the document 5. Sentence Permutation: the order of the sentence in the document was changed, in this case, the last sentence "I want to know" was changed to the second sentence to find the contextual information between sentences

thereby reducing the computational complexity associated with each task. As an illustrative example from the paper, a particular task might comprise a sequence of tokens, with each token designated to different experts.

Another significant advancement introduced in the Switch Transformer is the concept of "Simplified Sparse Routing". Traditionally, a single token would be routed to multiple 'k' experts, and the optimal response would be selected as the output to ensure both relevance and accuracy. Google streamlined this process. They posited that even if k equals 1, meaning each token is processed by just a single expert, the model can maintain its accuracy, all the while boosting efficiency.

Lastly, the model introduces "Efficient Sparse Routing". Google provided a formula to compute the optimal number of experts needed. This calculation takes into account the number of tokens in each batch and the number of experts active at a given stage. The "capacity factor" introduced in this mechanism serves as a buffer, permitting a surplus of experts to counter potential token overflows during the routing process.

5) GLM

GLM [12] introduced an innovative approach known as auto-regressive blank infilling to refine the masking and infilling techniques, setting it apart from models like BERT and T5. Instead of simply masking out tokens, GLM strategically removes continuous words from the input and then attempts to reconstruct them. A distinctive feature of GLM is that the prediction of masked regions can be permuted. Moreover, its positional embedding is designed in a 2-dimensional format. Impressively, with fewer parameters, GLM managed to outperform BERT on the SuperGLUE benchmark [314]. By leveraging Pattern Exploiting Training [315], GLM transformed various natural language understanding tasks into cloze-style problems, which could potentially have multiple correct answers.

Building upon the capabilities of GLM, ChatGLM [316]

was developed to provide conversational support.

Further expanding its applications, a multimodal version of GLM was introduced as VisualGLM [317]. This model supports visual conversations in both English and Chinese languages. Beyond the foundational architecture of GLM-6B, VisualGLM incorporates image data training based on the BLIP2-Qformer framework [318].

VII. PRE-TRAINING METHODS FOR LLMS

Pre-training is a critical phase in the development of Large Language Models (LLMs). This phase involves training the models on vast amounts of textual data to learn language patterns, structures, and contextual nuances. The effectiveness of pre-training significantly impacts the performance of LLMs on downstream tasks such as text generation, machine translation, summarization, and more. This section delves into various state-of-the-art pre-training methods for LLMs, categorized into different strategies such as training data reduction, neural architecture search, progressive learning, mixed precision training.

A. TRAINING DATA REDUCTION

Training data reduction techniques aim to minimize redundancy and improve the efficiency of the training process by selecting or augmenting the most relevant data.

- **COPA** [319]: Combining pre-training and adaptation strategies to enhance generalization.
- **MixMAE** [320]: Data augmentation and masking strategies to create diverse and challenging training examples.
- **Deduplicate Text Datasets** [321]: This method involves removing duplicate entries from the training data to reduce redundancy and improve training efficiency.
- **TRIPS** [322]: Task-aware pre-training data selection to ensure that the training data is relevant to the specific tasks the model will perform.
- **PatchDropout** [323]: Randomly dropping patches of input data to reduce computational requirements.
- **TPS** [324]: Token Pruning Strategy for efficient training by selectively pruning less important tokens.

B. NEURAL ARCHITECTURE SEARCH

Neural Architecture Search (NAS) involves automatically finding the best neural network architecture for a given task. These methods optimize the model design to achieve better performance.

- **PreNAS** [325]: Informed architecture search based on pre-training results.
- **PASHA** [326]: Progressive architecture search that evolves hybrid architectures over multiple stages.
- **ZICO** [327]: Zero-shot architecture search to identify optimal model structures without extensive training.
- **ElasticViT** [328]: Adaptive vision transformer architecture that adjusts computation based on input complexity.
- **RankNAS** [329]: Ranking neural architectures based on performance metrics .

C. PROGRESSIVE LEARNING

Progressive learning strategies involve training models in stages, gradually increasing the complexity and scale to improve performance and stability.

- **LiGO** [330]: Layerwise growth optimization to efficiently scale models.
- **Staged Training** [331]: Gradual increase in training complexity through multiple stages.
- **Knowledge Inheritance** [332]: Transferring knowledge progressively across model versions.
- **CompoundGrow** [333]: A strategy for progressively increasing model size during training.
- **stackingBERT** [334]: Stack-based training approach for incremental learning in BERT models.

D. MIXED PRECISION TRAINING

Mixed precision training techniques aim to balance training speed and model precision by using different numerical precisions for different parts of the model.

- **Mesa** [335]: Scheduling multi-epoch training with mixed precision adaptations.
- **GACT** [336]: Gradient accumulation with compression techniques to train large models efficiently.
- **blpa** [337]: Block-level precision adaptation for efficient training.
- **Mixture** [338]: Employing mixed precision training to enhance speed while maintaining accuracy.

The pre-training phase is vital for developing robust and efficient LLMs. The methods described represent recent advancements in pre-training, improving model performance, efficiency, and applicability. These strategies highlight the dynamic nature of NLP research. As LLMs evolve, exploring and implementing novel pre-training methods will be crucial for further advancements.

VIII. CHALLENGES OF LLMS

Large language models are the result of the development of neural networks and the technologies that followed such as like deep learning. For the modern cutting edge models, challenges still exist in the following phases:

- **Data Drawbacks**: Large language models requires a massive computational resource for the pre-training and fine-tuning.
- **Model Compression**: Large language models normally contains over billions of parameters, which cause memory intensive during both the training and deployment phase.
- **Distributed Computation**: Due to the increasing model size, some state-of-the-art large language models are trained on high performance clusters rather than local devices, which presents a challenge for distribution computation in the LLM field.
- **Multimodal Support**: Large language models can not only handle natural language processing tasks, but also

dealing with data from different format, that casues the multimodality support challenge of LLMs.

Cutting edge methods like Chain of Thought, Reinforcement Learning with Human Feedback, Transformer, and Mix of Expert have been proposed to train a model on a massive scale. A cursory review of the technologies that underpin the LLM technique will be provided in this section.

A. DATA DRAWBACKS

The use of massive datasets is one of the main characteristics of LLMs. These datasets are essential to the pre-training, fine-tuning, and evaluation processes of these models, as demonstrated by the experiment in [258], which demonstrates the superior performance of a smaller model trained with more labelled data than a larger model trained with less labelled data. However, as LLMs scale up, data challenges persist. The main points of the LLM problems pertaining to the data will be summarized in this section.

1) Quality of data

When it comes to relevance, richness, and redundancy, the quality of the data used to train LLMs is just as crucial as the model itself. Poor data quality can provide inaccurate and unreliable knowledge to the model as it learns from the datasets. The following factors could be the cause of the taxonomy of the data quality in this survey: (1. *Inaccurate Data*: The model will pick up problematic knowledge from these data, resulting to inaccurate or deceptive information in the model. (2. *Outdated Data*, this is particularly problematic in domains like technology and public events that are changing quickly. As an illustration, the most recent version of GPT-4, GPT-4 Turbo [339], used knowledge up to April 2023, but the original GPT-4 [191] used knowledge dated back to September 2021. Various technologies, including machine learning [340] and model editing [341]–[343], could be employed to mitigate this drawback. (3. *Redundancy of the data* describes the existence of redundant or overly similar information in the training dataset; if a dataset has multiple copies of the same content, this will lead to an overrepresentation of the model on the viewpoints, which will increase the model's biased understanding of particular topics. Certain measurements, like data deduplication [344], or the common solution from a range of state-of-the-art models, like Llama [10] and GPT [191], involve using data from many sources.

2) Bias of the data

The training data frequently involves biases in human languages or other forms of data input. These biases can span a wide range of topics, including gender, color, culture, religion, profession, and philosophy, along with preconceptions. As a result, concerns over justice and ethnicity may arise.The bias of language models was characterized by [345] from many perspectives with respect to the social impact. A few studies have tried to lessen the effects of language model bias. For example, [346] uses local edit to lessen gender bias, and [347] uses reinforced calibration to lessen political bias. [348] offered a novel way to measure the bias stereotype on these pretrianed models.

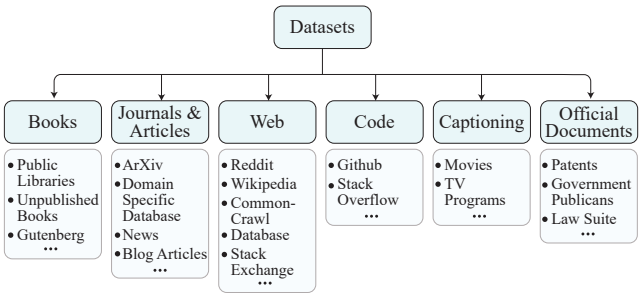

FIGURE 28. The common source of the dataset used for pre-training

3) Scale of Data

LLMs require massive amount of data to improve its accuracy and understanding of the prompts, which caused the challenge on the scale of the data regarding data collection, precessing and storage. The figure 15 shows the common source of the the datasets.

Following table shows the size of come popular datasets used by modern LLMs, single dataset such as The Pile and C4 already have hundreds of gigabytes and web-based database such as Common Crawl has already reached terabytes level.

Dataset Name	Category	Data Size
Common Crawl	Web	Terabytes level
RefinedWeb	Web	5 trillion tokens
The Pile	Diverse	800 GB
C4	Web	750 GB
Starcoder Data	Programming	783 GB
BookCorpus	Books	985 million words
ROOTS	Multilingual	1.6 TB
Wikipedia	Encyclopedia	19.88 GB
Red Pajama	Diverse	1.2 trillion tokens

TABLE 8. Open-Sourced Datasets for Training Large Language Models

Current cutting edge models normally applied datasets from different sources such as LLama [10] used up to 4.5T of the datasets and [5] applied 1.1T of the training data, which is also a huge challenge for the device used for the pre-training tasks.

B. MODEL COMPRESSION

In response to the space challenges posed by LLMs, this section provides a brief overview of three state-of-the-art model compression technologies. "Space challenges" in LLMs refer to the device memory limitations during pre-training, fine-tuning, and deployment due to the large size of model parameters. As LLMs grow larger to increase accuracy, more parameters are added, which intensifies computational demands. Model compression offers a solution by optimizing the internal structure of models to improve efficiency without significantly sacrificing

TABLE 9. Pruning techniques used in Large language models categorized in structured, unstructured, and their combination.

Structured	Sanh *et al.* [349], Cheong *et al.* [350], Gordon *et al.* [351], Wang *et al.* [352], Cui *et al.* [353], Yang *et al.* [354], Frantar *et al.* [355], Zhang *et al.* [356], Chen *et al.* [357], Sun *et al.* [358]
Unstructured	Xia *et al.* [359], Zhu *et al.* [360], Voita *et al.* [361], Fan *et al.* [362], Lagunas *et al.* [363], Michel *et al.* [364], Campos *et al.* [365], Santacroce *et al.* [366], Ma *et al.* [367], Guo *et al.* [368], Xia *et al.* [369]
Structured & Unstructured	Mishra *et al.* [370], Fang *et al.* [371], Holmes *et al.* [372], Fang *et al.* [373], Xu *et al.* [374]

performance. Three methods comprise the state-of-the-art model compression technique: (1) Pruning; (2) Quantization; and (3) Knowledge Distillation. These methods effectively address the efficiency and scalability challenges of LLMs by reducing model size and computational requirements.

1) Pruning

Pruning in large language models refers to the process of reducing model size by eliminating redundant and less critical structures. It can be categorized into two types: unstructured pruning and structured pruning. This optimization technique aims to shrink LLMs without significantly compromising their performance by selectively removing parameters considered less important for the model's task. Pruning methods eliminate redundant or non-informative parameters and can be performed in a structured or unstructured manner, each with its own strategies and impacts on model performance and efficiency. Typically, pruning involves a criterion to determine which weights to remove, based on factors like weight magnitude, training gradients, or other measures of importance. After pruning, the model usually undergoes fine-tuning to recover any lost performance due to parameter removal.

Structured pruning refers the pruning on the entire sets of the model structure such as channels, layers and weights. Structured pruning is advantageous for its compatibility with hardware optimization as it leads to a more regular and streamlined model structure with the trade-off of the substantial impact on the model's accuracy due to the removal of the entire structure.

Unstructured pruning only removes certain individual weights or nodes from a neural network with the least importance inside the model connection, which leads to a more fine-grained pruning results with significant reduction on the model size. Different from the structured pruning which has the regular structure, unstructured pruning is hard to be implemented on the hardware due to its irregularity.

Table 7 shows transformer based pruning technology that can be applied on large langauge models

2) Quantization

Quantization aims to decrease the memory footprint and computational demands of neural network models by converting high-precision model parameters, typically in extended data formats like 32-bit, into more compact representations, such as 8-bit formats, without significantly compromising performance. This technique is vital in model compression technologies and can be categorized into two main types: Post-Training Quantization (PTQ) [375] and Quantization Aware Training (QAT) [376].

In the context of LLMs, quantization reduces computational resource requirements by transforming model parameters from high to low precision, typically converting 32-bit floating-point weights and activations to 8-bit integer format. This approach benefits both the storage footprint and computation speed.

Quantization employs a graded approach, ranging from 3-bit quantization for the most compact model size to 8-bit for nearly full precision. Each increase in bit-size generally improves the model's accuracy but also increases its size and computational demands. The most aggressive 3-bit quantization combines different techniques for various parts of the model, while higher bit-sizes use more refined methods, allocating more bits to parts sensitive to precision loss. At the high end, 8-bit quantization closely approaches the model's original floating-point precision, yielding high accuracy at the expense of size and speed. This spectrum of quantization strategies allows flexible deployment of LLMs like LLaMA 2 across different use cases, balancing resource constraints and accuracy needs. Table 6 summarizes these quantization methods, detailing the techniques, bit sizes, and model sizes.

a: Post-Training Quantization

(PTQ) is a static quantization method applied after the model training process. It directly alters the original data format of the model without the need for additional data or modifications, aside from a few supplemental steps. In deep neural networks, the input typically adheres to a distinct pattern, facilitating statistical analysis. In PTQ, quantization algorithms convert the data format to a lower precision, guided by training data characteristics like the minimum and maximum weights and the distribution of activations. PTQ can be further subdivided into two methods: saturation and no saturation. The saturation approach employs KL divergence to identify a threshold T, which then rescales the data range. In contrast, the no saturation method determines the maximum value of the model weights and then maps this value to a more confined data format range.

b: Quantization Aware Training

In PTQ, quantization algorithms rely on statistical data from the model to determine the mapping, leading to a more significant discrepancy between the original and compressed models. On the other hand, Quantization Aware Training (QAT) adopts an online approach. Unlike the static methods in PTQ, QAT learns the scale and threshold during the training process by simulating the quantization effects. During QAT, a scaling ratio is established to map intermediate values. By allowing quantization to be back-propagated, this method

Quant Method	Bits	Size	Description
q3_K	3	2.95 GB	New k-quant method for all tensors, moderate size and RAM requirements.
q3_K_M	3	3.28 GB	A variation of k-quant applying different bits for attention and feed-forward tensors.
q3_K_L	3	3.60 GB	K-quant with higher bit allocation for select attention and feed-forward tensors.
q4_0	4	3.79 GB	Original quant method with uniform 4-bit allocation across all tensors.
q5_0	5	4.63 GB	Original 5-bit quant method for higher accuracy at the cost of increased resource usage.
q6_K	6	5.53 GB	New k-quant method using 6-bit quantization for all tensors, a balance between precision and size.
q8_0	8	7.16 GB	8-bit quantization offering high accuracy, suitable for scenarios where resource constraints are minimal.

TABLE 10. Summary of different quantization methods for the LLaMA 2 model with a focus on k-quant strategies and resource implications.

TABLE 11. Quantization algorithms using QAT and PTQ

QAT	LLM-QAT [377], PEQA [378], QLORA [379]
PTQ	GPTQ [380], OPTQ [381], RPTQ [382], FPTQ [383], ZeroQuant [384], ZeroQuant-v2 [385], ZeroQuant-FP [386], SmoothQuant [387], OmniQuant [388], OWQ [389], AWQ [390], LLM.int8() [391], W4A4 [392], ResQ [393], SqueezeLLM [394], QUIP [395], SignRound [396], Norm Tweaking [397], OLiVe [398], QuantEase [399], Outlier Suppresssion [400], Outlier Suppresssion+ [401], LUT-GEMM [402]

results in a reduced quantization loss, making the quantized weights more akin to the original model's weights.

3) Knowledge Distillation

Knowledge distillation in large language models (LLMs) is a technique aimed at streamlining their vast knowledge into more compact and efficient forms. This process involves training a smaller, student model to emulate the behavior of a larger, teacher model, effectively transferring the sophisticated decision-making abilities of LLMs to smaller models. This makes the smaller models suitable for environments with limited computational resources, maintaining core functionalities while significantly reducing computational overhead.

Knowledge Distillation (KD) compresses the knowledge of a larger, more complex teacher model into a smaller, more efficient student model. This allows the student model to perform at a level close to the teacher but with a fraction of the computational requirements. Two main approaches, White-Box and Black-Box KD, are used in this process.

a: White-Box Knowledge Distillation

This method involves using not only the outputs of the teacher model but also its internal representations and states to guide the training of the student model. This richer transfer of knowledge provides the student with insights into the intermediate processing of the teacher.

b: Black-Box Knowledge Distillation

In contrast, Black-Box KD uses only the final outputs of the teacher model. The student learns to mimic the teacher's output distribution without access to its internal workings, making this method more flexible as it doesn't require the student's architecture to match the teacher's internal structure.

The key challenge in both types of Knowledge Distillation (KD) is transferring as much relevant information as possible from the teacher to the student model. This often involves training the student to reproduce the teacher's output probabilities, which carry more information than just the final predicted class. The effectiveness of KD can be measured by how well the student model performs compared to the teacher on a set of tasks, ideally achieving similar performance while being more efficient to run.

Advancements in this domain have introduced innovative strategies to enhance the student model's learning process, such as selecting the most informative elements from the teacher model's knowledge and utilizing intermediate representations for a richer training experience. These techniques ensure that the distilled model replicates the critical aspects of the teacher model's performance. The impact of knowledge distillation extends beyond model efficiency, enabling the deployment of advanced language processing tools in diverse applications and promoting sustainable AI practices by reducing computational and energy demands.

C. DISTRIBUTION COMPUTATION

Owing to the immense scale of large language models, which can reach up to trillions of parameters, traditional deep learning methods using single-device training or deployment are insufficient to handle the vast datasets and expansive parameter sizes associated with these models. As a result, distributed computation has emerged as a pivotal solution. Presently, three primary distributed computation methods are employed to address these challenges: data parallelism enhances the speed of model training, while tensor parallelism and pipeline parallelism enable the training of models that exceed the device's memory capacity.

1) Tensor parallel

The fundamental concept of tensor parallelism involves dividing the entire tensor of a model into distinct segments. Each device retains one segment of the tensor, and the final results can be procured by concatenating these tensor segments based on the dimensions from which they were partitioned. As an intra-layer parallelism method, its primary advantage is the ability to obtain results through a singular concatenation operation. However, a drawback of tensor

parallelism is the necessity for an additional step to ensure the accuracy of the concatenation.

2) Pipeline parallel

Pipeline parallelism involves segmenting the entire model into multiple sections based on its layers, a method also referred to as inter-layer parallelism. In this approach, each device manages several layers of the model, and data is sequentially processed through the devices in accordance with the model's structure. Unlike tensor parallelism, pipeline parallelism doesn't require additional operations, as each device contains a complete segment of the model. However, a limitation is that devices must await data output from the preceding device, leading to idle periods. Consequently, pipeline parallelism can result in the underutilization of computational resources.

3) Data parallel

Unlike the previously mentioned approaches, which focus on accommodating models larger than what a single device can handle, the objective of data parallelism is to expedite the training process by harnessing computational power from multiple devices. In data parallelism, every device retains a copy of the model and is assigned a distinct data batch. This setup facilitates parallelized training, thereby enhancing the training speed. However, because each device has to store a complete replica of the model, a notable drawback of data parallelism is the inefficient use of device memory.

D. MULTIMODAL SUPPORT

A significant challenge for contemporary Large Language Models (LLMs) is supporting multimodality, especially since the advent of the Vision Transformer (ViT) [214], which showcased the potential of transformers for visual tasks. Unlike conventional LLMs, training models with multimodal support is more intricate due to the need for aligning representations across different modalities. This introduces distinct training tasks for these multimodal LLMs. This section is structured based on these tasks, which are categorized into pre-training tasks and downstream tasks.

1) Image-text matching

Image-Text Matching (ITM) is a method that aligns data from different modalities from a coarse-grained perspective. The primary objective of ITM is to predict the relationship between two segments, typically an image-captioning pair. This enables the model to learn the representation of text and its corresponding images. ITM has been extensively employed in state-of-the-art models. Examples include VILBERT [403], B2T2 [404] — which employs bounding boxes to fuse image patches with textual information for enhanced visual-text integration — as well as LXMERT [405], XLXMERT [406], VisualBERT [167], UNITER [407], Unicoder-VL [408], Pixel-BERT [409], ERNIE-VIL [176], ERNIE-VIL 2.0 [177], and UNIMO [410], [411]. Furthermore, the BLIP series of models [paper, paper, paper] also incorporated ITM as one of their training tasks.

2) Cross-modal contrastive learning

In the Image-Text Matching (ITM) task, the model typically determines whether a visual-text pair's information aligns. Meanwhile, Cross-Modal Contrastive Learning (CMCL) seeks to enhance the association between image and text pairs based on their similarity. More specifically, CMCL operates like a clustering task, aiming to differentiate unrelated visual-text pairs and cluster closely related ones. The study [Leveraging Visual Knowledge in Language Tasks] explored the efficacy of the CMCL task. Several models employed this strategy, including UNIMO [410], UNIMO2 [410], WudaoMM [412], Taisu [413], CLIP [270], CLIP 2 [414], and ALIGN [278].

3) Cross-modal masked language matching

Cross-modal masked language modeling (MLM) draws parallels to BERT by masking a portion of the input data, prompting the model to predict it during training. This straightforward approach is particularly effective for semantically dependent data. It's one of the earliest strategies introduced by ViT and has become popular in LLMs with multimodal support due to its adaptability. Numerous multimodal large language models, such as VisualBERT [167], ViLT [415], and InterBERT [416], incorporate MLM in their training processes.

4) Masked region modeling

Unlike Masked Language Modeling (MLM) which masks textual information, Masked Region Modeling (MRM) is designed for visual input masking tasks and can be categorized into two main approaches: classification and regression.

Masked Region Classification (MRC) has its origins in the Masked Region Prediction task (MRP). However, in MRC, masking is applied to an entire region of interest rather than lower-level tokens like patches or pixels. The objective is to predict the higher-level semantics of multimodal input by determining the classification of the masked region, using visible regions as context. Much like the ubiquity of MLM in text-based models, many multimodal large language models adopt MRC during training, notable examples being VL-BERT and Unicoder-VL. While the conventional MRC task utilizes cross-entropy, some models, such as UNIMO, have incorporated a variant known as MRC-kl, which employs KL-divergence as introduced by the UNITER framework.

Masked Region Feature Regression (MRSR), another offspring of MRP, employs regression techniques to reconstruct the feature map, rather than classifying masked regions. Models like ImageBERT [417] and UNITER have incorporated MRSR into their training paradigms.

5) Other pre-training tasks

Except the tasks mentioned above, there is other tasks used by either pre-training tasks and downstream tasks on LLMs,

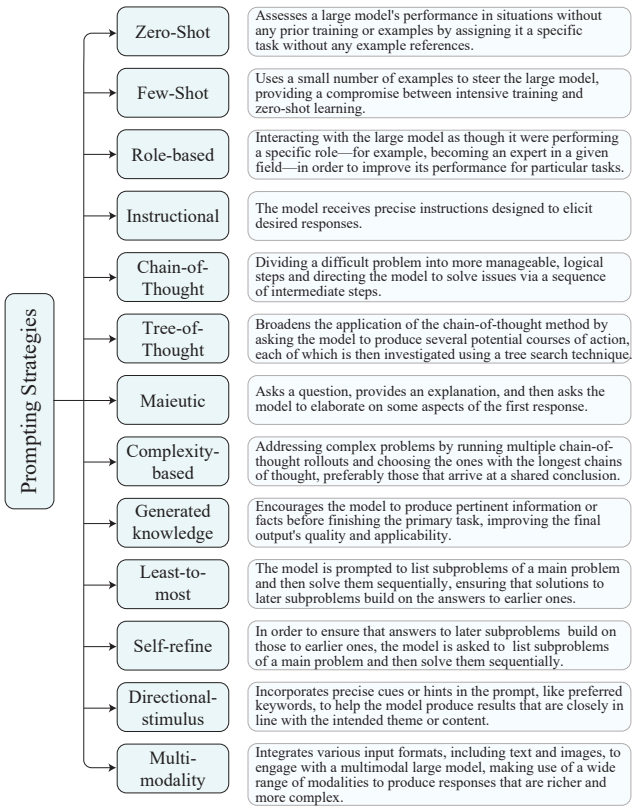

FIGURE 29. The most commonly utilized Prompt Engineering strategies.

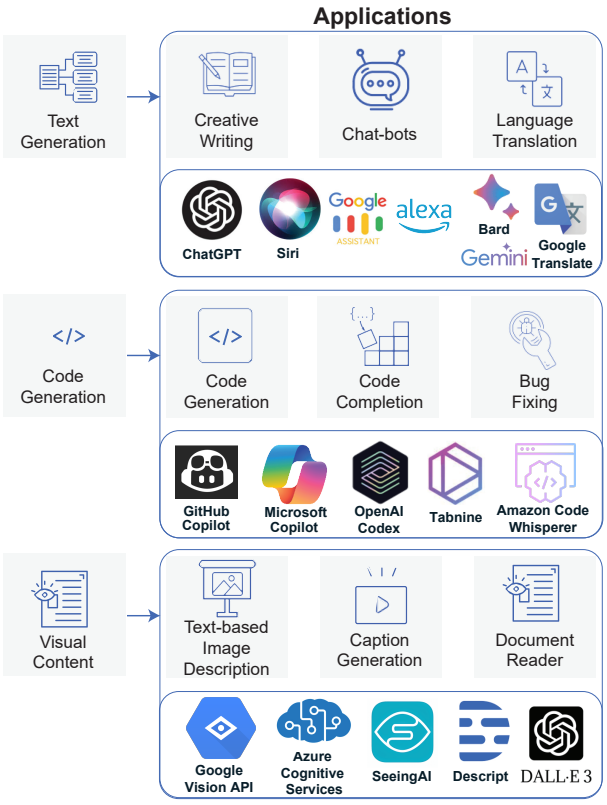

FIGURE 30. Applications of LLMs and MLLMs across various domains including text generation, code generation, and visual content. These applications showcase the versatility and impact of large language models in enhancing productivity and innovation.

which is not as commonly used compared with the previous mentioned models but were used by some specific models

1) WRA: Word Region Alignment
2) Seq2Seq: Sequence to Sequence generation
3) VQA: Visual Question Answering
4) MRFR: Masked Region Feature Regression
5) SGP: Scene Graph Prediction
6) VLC: Vision-Language Contrastive Learning
7) MTL: Multi Task Learning
8) WPA: Word-Patch Alignment
9) MVM (MFC): Masked Vision Modeling
10) VLM: Vision-Language Matching
11) ITC: Image-Text Contrastive
12) ITG: Image-Text Generation
13) PrefixLM: Prefix Language Modeling
14) MOC: Masked Object Classification

E. PROMPT ENGINEERING

Another essential technique that speeds up the comprehension of LLMs in context is prompt engineering, which strategically formulates input queries that include content and instruction. This technique is simpler than pre-training and fine-tuning and allows users to interact with the LLM to control the token datastream.

The following advantages arise from using prompt engineering during LLM inference. Prompt strategies reduce human bias in training data for LLMs by making it easier for users to find relevant results. This is primarily because of the interaction between the user and the system. Second, the model users' prompts have a high information density when compared to the training data used for pre-training and fine-tuning, suggesting that prompting is worth hundreds of data points on average ([418]). Thirdly, prompt engineering can be customized so that users can achieve excellent accuracy performance even in the absence of training, particularly for downstream tasks. This feature offers prompt engineering unparalleled benefits in addition to training phases. Figure 29 is a taxonomy of some popular prompting techniques.

IX. APPLICATIONS OF LLMS

LLMs have revolutionized various domains by leveraging their capabilities to understand, generate, and manipulate human language. Their applications span a wide array of fields, including visual content creation, audio generation, text generation, code generation, and design automation. While LLMs primarily handle text, they are often integrated with other systems to process and generate multimedia content. The versatility and efficacy of LLMs have led to significant advancements in these areas, with numerous companies developing notable products to harness their potential.

A. TEXT GENERATION

Text generation is one of the most prominent applications of LLMs. It encompasses several tasks such as creative writing, chat-bots, and language translation.

- **Creative Writing**: LLMs based tools like OpenAI's GPT-4 and -4o, Bard, Gemini and other writing assistants help authors generate content, brainstorm ideas, and overcome writer's block.
- **Chat-bots**: Companies like Google, Apple, and Amazon utilize LLMs in their virtual assistants (Google Assistant, Siri, Alexa) to provide intelligent and conversational responses, improving user interaction.
- **Language Translation**: Tools like Google Translate and Microsoft's translation services leverage LLMs to provide accurate and context-aware translations across multiple languages.

B. CODE GENERATION

LLMs have also made significant strides in the field of code generation, aiding developers in writing and optimizing code.

- **Code Generation**: Products like GitHub Copilot and OpenAI Codex assist developers by generating code snippets and automating repetitive tasks, thus speeding up the development process.
- **Code Completion**: Tools such as Tabnine and Microsoft Copilot offer real-time code suggestions and completions, enhancing coding efficiency and reducing errors.
- **Bug Fixing**: Services like Amazon CodeWhisperer and other AI-powered code analysis tools help in identifying and fixing bugs, improving the overall quality of the software.

C. VISUAL CONTENT UNDERSTANDING

In addition to text and code, LLMs are instrumental in understanding visual content. This includes tasks like text-based image description, caption generation, and document reading through OCR (Optical Character Recognition).

- **Text-based Image Description**: Tools like Google Vision API and Azure Cognitive Services use LLMs to provide detailed descriptions of images, enhancing accessibility for visually impaired users.
- **Caption Generation**: Applications like SeeingAI and DALL-E 3 generate captions for images, making content more accessible and searchable.
- **Document Reader**: Products such as Descript and other OCR technologies convert scanned documents and images into machine-readable text, facilitating easier access and analysis.

The applications of LLMs across various domains demonstrate their vast potential in enhancing productivity and fostering innovation. As these models continue to evolve, we can expect even more groundbreaking applications that will further transform the way we interact with and utilize technology.

X. CONCLUSION

This paper offers an exhaustive review of Large Language Models (LLMs) and their evolution within the domain of Natural Language Processing (NLP). It explores the diverse proficiencies of LLMs across various NLP tasks, including text generation, logical reasoning, machine translation, summarization, and multimodal integration. LLMs are systematically categorized into three primary architectures: encoder-only, decoder-only, and encoder-decoder frameworks. Additionally, the paper highlights the inherent challenges and constraints of LLMs, notably their dependency on statistical patterns as opposed to genuine understanding, and showcases state-of-the-art methodologies in pre-training and fine-tuning. The paper also discusses various model compression techniques, benchmarks, and applications. Overall, this paper offers valuable insights into the capabilities, challenges, and future prospects of LLMs in NLP applications.

ACKNOWLEDGEMENT

This work was partially supported by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010, and the NYUAD Center for CyberSecurity (CCS), funded by Tamkeen under the NYUAD Research Institute Award G1104.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
[2] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.
[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[4] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. Wang, X. Zhang *et al.*, "Pangu-{\alpha}: Large-scale autoregressive pretrained chinese language models with auto-parallel computation," *arXiv preprint arXiv:2104.12369*, 2021.
[5] X. Ren, P. Zhou, X. Meng, X. Huang, Y. Wang, W. Wang, P. Li, X. Zhang, A. Podolskiy, G. Arshinov *et al.*, "Pangu-{\Sigma}: Towards trillion parameter language model with sparse heterogeneous computing," *arXiv preprint arXiv:2303.10845*, 2023.
[6] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.
[7] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, no. 64-67, p. 2, 2001.
[8] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," *arXiv preprint arXiv:1905.07129*, 2019.
[9] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.
[10] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.
[11] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.
[12] Z. Du, Y. Qian, X. Liu, M. Ding, J. Qiu, Z. Yang, and J. Tang, "Glm: General language model pretraining with autoregressive blank infilling," *arXiv preprint arXiv:2103.10360*, 2021.

[13] D. P. Kingma, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE, 2018, pp. 415–419.

[15] R. Wei, C. Garcia, A. El-Sayed, V. Peterson, and A. Mahmood, "Variations in variational autoencoders-a comparative evaluation," *Ieee Access*, vol. 8, pp. 153 651–153 670, 2020.

[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Science Robotics*, vol. 3, pp. 2672–2680, 6 2014.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 11 2015.

[18] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint: arXiv:1411.1784*, 11 2014.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint: arXiv:1701.07875*, 1 2017.

[20] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *Advances in Neural Information Processing Systems*, vol. 2017-December, pp. 5768–5778, 3 2017.

[21] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 2813–2821, 11 2016.

[22] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 2242–2251, 3 2017.

[23] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 4217–4228, 12 2018.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 12 744–12 753, 5 2018.

[25] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *7th International Conference on Learning Representations, ICLR 2019*, 9 2018.

[26] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 10 2017.

[27] Y. Choi, M. Choi, M. Kim, J. W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797, 11 2017.

[28] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys*, vol. 54, 6 2019.

[29] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye, "A review on generative adversarial networks: Algorithms, theory, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, pp. 3313–3332, 4 2023.

[30] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, pp. 53–65, 10 2017.

[31] I. O. Gallegos, R. A. Rossi, J. Barrow, M. M. Tanjim, S. Kim, F. Dernoncourt, T. Yu, R. Zhang, and N. K. Ahmed, "Bias and fairness in large language models: A survey," *arXiv preprint arXiv:2309.00770*, 2023.

[32] H. Shi, Z. Xu, H. Wang, W. Qin, W. Wang, Y. Wang, and H. Wang, "Continual learning of large language models: A comprehensive survey," *arXiv preprint arXiv:2404.16789*, 2024.

[33] Z. Zhou, X. Ning, K. Hong, T. Fu, J. Xu, S. Li, Y. Lou, L. Wang, Z. Yuan, X. Li *et al.*, "A survey on efficient inference for large language models," *arXiv preprint arXiv:2404.14294*, 2024.

[34] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, and C. Faloutsos, "Large language models (llms) on tabular data: Prediction, generation, and understanding–a survey," *arXiv preprint arXiv:2402.17944*, 2024.

[35] B. C. Das, M. H. Amini, and Y. Wu, "Security and privacy challenges of large language models: A survey," *arXiv preprint arXiv:2402.00888*, 2024.

[36] H. Jin, L. Hu, X. Li, P. Zhang, C. Chen, J. Zhuang, and H. Wang, "Jailbreakzoo: Survey, landscapes, and horizons in jailbreaking large language and vision-language models," *arXiv preprint arXiv:2407.01599*, 2024.

[37] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu *et al.*, "A survey on large language models for recommendation," *arXiv preprint arXiv:2305.19860*, 2023.

[38] M. Akyash and H. M. Kamali, "Evolutionary large language models for hardware security: A comparative survey," *arXiv preprint arXiv:2404.16651*, 2024.

[39] T. Bai, H. Liang, B. Wan, L. Yang, B. Li, Y. Wang, B. Cui, C. He, B. Yuan, and W. Zhang, "A survey of multimodal large language model from a data-centric perspective," *arXiv preprint arXiv:2405.16640*, 2024.

[40] H. Xiao, F. Zhou, X. Liu, T. Liu, Z. Li, X. Liu, and X. Huang, "A comprehensive survey of large language models and multimodal large language models in medicine," *arXiv preprint arXiv:2405.08603*, 2024.

[41] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu *et al.*, "Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *arXiv preprint arXiv:2405.10825*, 2024.

[42] Y. Huang, K. Tang, and M. Chen, "A comprehensive survey on evaluating large language model applications in the medical industry," *arXiv preprint arXiv:2404.15777*, 2024.

[43] C. Qu, S. Dai, X. Wei, H. Cai, S. Wang, D. Yin, J. Xu, and J.-R. Wen, "Tool learning with large language models: A survey," *arXiv preprint arXiv:2405.17935*, 2024.

[44] L. Qin, Q. Chen, X. Feng, Y. Wu, Y. Zhang, Y. Li, M. Li, W. Che, and P. S. Yu, "Large language models meet nlp: A survey," *arXiv preprint arXiv:2405.12819*, 2024.

[45] Z. Zhang, Y. Sun, Z. Wang, Y. Nie, X. Ma, P. Sun, and R. Li, "Large language models for mobility in transportation systems: A survey on forecasting tasks," *arXiv preprint arXiv:2405.02357*, 2024.

[46] S. Kukreja, T. Kumar, A. Purohit, A. Dasgupta, and D. Guha, "A literature survey on open source large language models," in *Proceedings of the 2024 7th International Conference on Computers in Management and Business*, 2024, pp. 133–143.

[47] L. Qin, Q. Chen, Y. Zhou, Z. Chen, Y. Li, L. Liao, M. Li, W. Che, and P. S. Yu, "Multilingual large language model: A survey of resources, taxonomy and frontiers," *arXiv preprint arXiv:2404.04925*, 2024.

[48] S. Dai, C. Xu, S. Xu, L. Pang, Z. Dong, and J. Xu, "Unifying bias and unfairness in information retrieval: A survey of challenges and opportunities with large language models," *arXiv preprint arXiv:2404.11457*, 2024.

[49] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen, "A survey on multimodal large language models," *arXiv preprint arXiv:2306.13549*, 2023.

[50] Z. Zhang, X. Bo, C. Ma, R. Li, X. Chen, Q. Dai, J. Zhu, Z. Dong, and J.-R. Wen, "A survey on the memory mechanism of large language model based agents," *arXiv preprint arXiv:2404.13501*, 2024.

[51] S. Hu, T. Huang, F. Ilhan, S. Tekin, G. Liu, R. Kompella, and L. Liu, "A survey on large language model-based game agents," *arXiv preprint arXiv:2404.02039*, 2024.

[52] Z. Bai, P. Wang, T. Xiao, T. He, Z. Han, Z. Zhang, and M. Z. Shou, "Hallucination of multimodal large language models: A survey," *arXiv preprint arXiv:2404.18930*, 2024.

[53] Y. Huang and J. Huang, "A survey on retrieval-augmented text generation for large language models," *arXiv preprint arXiv:2404.10981*, 2024.

[54] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "Pre-trained language models for text generation: A survey," *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–39, 2024.

[55] Y. Liu, Y. Yao, J.-F. Ton, X. Zhang, R. G. H. Cheng, Y. Klochkov, M. F. Taufiq, and H. Li, "Trustworthy llms: a survey and guideline for evaluating large language models' alignment," *arXiv preprint arXiv:2308.05374*, 2023.

[56] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A survey on large language model (llm) security and privacy: The good, the bad, and the ugly," *High-Confidence Computing*, p. 100211, 2024.

[57] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang *et al.*, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024.

[58] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[59] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu *et al.*, "Instruction tuning for large language models: A survey," *arXiv preprint arXiv:2308.10792*, 2023.
[60] R. Hong, X. Pang, and C. Zhang, "Advances in reasoning by prompting large language models: A survey," *Cybernetics and Intelligence*, 2024.
[61] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, and X. Cheng, "On protecting the data privacy of large language models (llms): A survey," *arXiv preprint arXiv:2403.05156*, 2024.
[62] Y. Cao, H. Zhao, Y. Cheng, T. Shu, G. Liu, G. Liang, J. Zhao, and Y. Li, "Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods," *arXiv preprint arXiv:2404.00282*, 2024.
[63] X. Liu, P. Xu, J. Wu, J. Yuan, Y. Yang, Y. Zhou, F. Liu, T. Guan, H. Wang, T. Yu *et al.*, "Large language models and causal inference in collaboration: A comprehensive survey," *arXiv preprint arXiv:2403.09606*, 2024.
[64] A. Esmradi, D. W. Yip, and C. F. Chan, "A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models," in *International Conference on Ubiquitous Security*. Springer, 2023, pp. 76–95.
[65] A. G. Chowdhury, M. M. Islam, V. Kumar, F. H. Shezan, V. Jain, and A. Chadha, "Breaking down the defenses: A comparative survey of attacks on large language models," *arXiv preprint arXiv:2403.04786*, 2024.
[66] Z. Sun, "A short survey of viewing large language models in legal aspect," *arXiv preprint arXiv:2303.09136*, 2023.
[67] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, "Explainability for large language models: A survey," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 2, pp. 1–38, 2024.
[68] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," *arXiv preprint arXiv:2308.07107*, 2023.
[69] J. Li, Y. Liu, C. Liu, L. Shi, X. Ren, Y. Zheng, Y. Liu, and Y. Xue, "A cross-language investigation into jailbreak attacks in large language models," *arXiv preprint arXiv:2401.16765*, 2024.
[70] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou *et al.*, "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023.
[71] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *arXiv preprint arXiv:2311.05232*, 2023.
[72] E. Shayegani, M. A. A. Mamun, Y. Fu, P. Zaree, Y. Dong, and N. Abu-Ghazaleh, "Survey of vulnerabilities in large language models revealed by adversarial attacks," *arXiv preprint arXiv:2310.10844*, 2023.
[73] H. Zhang, H. Song, S. Li, M. Zhou, and D. Song, "A survey of controllable text generation using transformer-based pre-trained language models," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–37, 2023.
[74] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, "Recent advances in natural language processing via large pre-trained language models: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–40, 2023.
[75] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen *et al.*, "Siren's song in the ai ocean: a survey on hallucination in large language models," *arXiv preprint arXiv:2309.01219*, 2023.
[76] X. Zhu, J. Li, Y. Liu, C. Ma, and W. Wang, "A survey on model compression for large language models," *arXiv preprint arXiv:2308.07633*, 2023.
[77] L. Hu, Z. Liu, Z. Zhao, L. Hou, L. Nie, and J. Li, "A survey of knowledge enhanced pre-trained language models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 4, pp. 1413–1430, 2024.
[78] B. Wang, Q. Xie, J. Pei, Z. Chen, P. Tiwari, Z. Li, and J. Fu, "Pre-trained language models in biomedical domain: A systematic survey," *ACM Computing Surveys*, vol. 56, no. 3, pp. 1–52, 2023.
[79] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," *arXiv preprint arXiv:2212.10403*, 2022.
[80] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier *et al.*, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and individual differences*, vol. 103, p. 102274, 2023.
[81] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.
[82] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz *et al.*, "Augmented language models: a survey," *arXiv preprint arXiv:2302.07842*, 2023.
[83] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "Ammu: a survey of transformer-based biomedical pretrained language models," *Journal of biomedical informatics*, vol. 126, p. 103982, 2022.
[84] ——, "Ammus: A survey of transformer-based pretrained models in natural language processing," *arXiv preprint arXiv:2108.05542*, 2021.
[85] M. Zaib, Q. Z. Sheng, and W. Emma Zhang, "A short survey of pre-trained language models for conversational ai-a new age in nlp," in *Proceedings of the Australasian computer science week multiconference*, 2020, pp. 1–4.
[86] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.
[87] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," *arXiv preprint arXiv:2110.04366*, 2021.
[88] Y. Zhu, J. Feng, C. Zhao, M. Wang, and L. Li, "Counter-interference adapter for multilingual machine translation," *arXiv preprint arXiv:2104.08154*, 2021.
[89] T. Lei, J. Bai, S. Brahma, J. Ainslie, K. Lee, Y. Zhou, N. Du, V. Y. Zhao, Y. Wu, B. Li *et al.*, "Conditional adapters: Parameter-efficient transfer learning with fast inference," *arXiv preprint arXiv:2304.04947*, 2023.
[90] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "Adapterfusion: Non-destructive task composition for transfer learning," *arXiv preprint arXiv:2005.00247*, 2020.
[91] Y. Wang, S. Mukherjee, X. Liu, J. Gao, A. H. Awadallah, and J. Gao, "Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models," *arXiv preprint arXiv:2205.12410*, vol. 1, no. 2, p. 4, 2022.
[92] H. Zhao, J. Fu, and Z. He, "Prototype-based hyperadapter for sample-efficient multi-task tuning," *arXiv preprint arXiv:2310.11670*, 2023.
[93] A. Chronopoulou, M. E. Peters, A. Fraser, and J. Dodge, "Adaptersoup: Weight averaging to improve generalization of pretrained language models," *arXiv preprint arXiv:2302.07027*, 2023.
[94] S. He, R.-Z. Fan, L. Ding, L. Shen, T. Zhou, and D. Tao, "Mera: Merging pretrained adapters for few-shot learning," *arXiv preprint arXiv:2308.15982*, 2023.
[95] R. K. Mahabadi, S. Ruder, M. Dehghani, and J. Henderson, "Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks," *arXiv preprint arXiv:2106.04489*, 2021.
[96] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.
[97] J. Li, W. Aitken, R. Bhambhoria, and X. Zhu, "Prefix propagation: Parameter-efficient tuning for long sequences," *arXiv preprint arXiv:2305.12086*, 2023.
[98] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *arXiv preprint arXiv:2110.07602*, 2021.
[99] Z.-R. Zhang, C. Tan, H. Xu, C. Wang, J. Huang, and S. Huang, "Towards adaptive prefix tuning for parameter-efficient language model fine-tuning," *arXiv preprint arXiv:2305.15212*, 2023.
[100] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv preprint arXiv:2103.10385*, 2021.
[101] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.
[102] F. Ma, C. Zhang, L. Ren, J. Wang, Q. Wang, W. Wu, X. Quan, and D. Song, "Xprompt: Exploring the extreme of prompt tuning," *arXiv preprint arXiv:2210.04457*, 2022.
[103] Z. Wu, S. Wang, J. Gu, R. Hou, Y. Dong, V. Vydiswaran, and H. Ma, "Idpg: An instance-dependent prompt generation method," *arXiv preprint arXiv:2204.04497*, 2022.
[104] X. Liu, T. Sun, X. Huang, and X. Qiu, "Late prompt tuning: A late prompt could be better than many prompts," *arXiv preprint arXiv:2210.11292*, 2022.
[105] W. Zhu and M. Tan, "Spt: Learning to selectively insert prompts for better prompt tuning," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 11 862–11 878.
[106] Q. Wang, Y. Mao, J. Wang, H. Yu, S. Nie, S. Wang, F. Feng, L. Huang, X. Quan, Z. Xu *et al.*, "Aprompt: Attention prompt tuning for efficient adaptation of pre-trained language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 9147–9160.

[107] T. Vu, B. Lester, N. Constant, R. Al-Rfou, and D. Cer, "Spot: Better frozen model adaptation through soft prompt transfer," *arXiv preprint arXiv:2110.07904*, 2021.

[108] Y. Su, X. Wang, Y. Qin, C.-M. Chan, Y. Lin, H. Wang, K. Wen, Z. Liu, P. Li, J. Li *et al.*, "On transferability of prompt tuning for natural language processing," *arXiv preprint arXiv:2111.06719*, 2021.

[109] J. Wu, T. Yu, R. Wang, Z. Song, R. Zhang, H. Zhao, C. Lu, S. Li, and R. Henao, "Infoprompt: Information-theoretic soft prompt tuning for natural language understanding," *arXiv preprint arXiv:2306.04933*, 2023.

[110] L. Chen, H. Huang, and M. Cheng, "Ptp: Boosting stability and performance of prompt tuning with perturbation-based regularizer," *arXiv preprint arXiv:2305.02423*, 2023.

[111] Y. Qin, X. Wang, Y. Su, Y. Lin, N. Ding, J. Yi, W. Chen, Z. Liu, J. Li, L. Hou *et al.*, "Exploring universal intrinsic task subspace via prompt tuning," *arXiv preprint arXiv:2110.07867*, 2021.

[112] J.-Y. Choi, J. Kim, J.-H. Park, W.-L. Mok, and S. Lee, "Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 14 306–14 316.

[113] Z. Shi and A. Lipani, "Dept: Decomposed prompt tuning for parameter-efficient fine-tuning," *arXiv preprint arXiv:2309.05173*, 2023.

[114] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. A. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1950–1965, 2022.

[115] T. Zadouri, A. Üstün, A. Ahmadian, B. Ermiş, A. Locatelli, and S. Hooker, "Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning," *arXiv preprint arXiv:2309.05444*, 2023.

[116] D. Lian, D. Zhou, J. Feng, and X. Wang, "Scaling & shifting your features: A new baseline for efficient model tuning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 109–123, 2022.

[117] X. Lu, F. Brahman, P. West, J. Jang, K. Chandu, A. Ravichander, L. Qin, P. Ammanabrolu, L. Jiang, S. Ramnath *et al.*, "Inference-time policy adapters (ipa): Tailoring extreme-scale lms without fine-tuning," *arXiv preprint arXiv:2305.15065*, 2023.

[118] D. Guo, A. M. Rush, and Y. Kim, "Parameter-efficient transfer learning with diff pruning," *arXiv preprint arXiv:2012.07463*, 2020.

[119] N. Lawton, A. Kumar, G. Thattai, A. Galstyan, and G. V. Steeg, "Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models," *arXiv preprint arXiv:2305.16597*, 2023.

[120] B. Liao, Y. Meng, and C. Monz, "Parameter-efficient fine-tuning without introducing new latency," *arXiv preprint arXiv:2305.16742*, 2023.

[121] Y.-L. Sung, V. Nair, and C. A. Raffel, "Training neural networks with fixed sparse masks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 193–24 205, 2021.

[122] S. S. S. Das, R. H. Zhang, P. Shi, W. Yin, and R. Zhang, "Unified low-resource sequence labeling by sample-aware dynamic sparse finetuning," *arXiv preprint arXiv:2311.03748*, 2023.

[123] A. Ansell, E. M. Ponti, A. Korhonen, and I. Vulić, "Composable sparse fine-tuning for cross-lingual transfer," *arXiv preprint arXiv:2110.07560*, 2021.

[124] Z. Fu, H. Yang, A. M.-C. So, W. Lam, L. Bing, and N. Collier, "On the effectiveness of parameter-efficient fine-tuning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 12 799–12 807.

[125] R. Xu, F. Luo, Z. Zhang, C. Tan, B. Chang, S. Huang, and F. Huang, "Raise a child in large language model: Towards effective and generalizable fine-tuning," *arXiv preprint arXiv:2109.05687*, 2021.

[126] D. Vucetic, M. Tayaranian, M. Ziaeefard, J. J. Clark, B. H. Meyer, and W. J. Gross, "Efficient fine-tuning of bert models on the edge," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 1838–1842.

[127] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," *arXiv preprint arXiv:2106.10199*, 2021.

[128] M. Gheini, X. Ren, and J. May, "Cross-attention is all you need: Adapting pretrained transformers for machine translation," *arXiv preprint arXiv:2104.08771*, 2021.

[129] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," *arXiv preprint arXiv:2012.13255*, 2020.

[130] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[131] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1022–1035, 2021.

[132] A. Edalati, M. Tahaei, I. Kobyzev, V. P. Nia, J. J. Clark, and M. Rezagholizadeh, "Krona: Parameter efficient tuning with kronecker adapter," *arXiv preprint arXiv:2212.10650*, 2022.

[133] X. He, C. Li, P. Zhang, J. Yang, and X. E. Wang, "Parameter-efficient model adaptation for vision transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 817–825.

[134] D. J. Kopiczko, T. Blankevoort, and Y. M. Asano, "Vera: Vector-based random matrix adaptation," *arXiv preprint arXiv:2310.11454*, 2023.

[135] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen, "Dora: Weight-decomposed low-rank adaptation," *arXiv preprint arXiv:2402.09353*, 2024.

[136] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi, "Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation," *arXiv preprint arXiv:2210.07558*, 2022.

[137] Q. Zhang, M. Chen, A. Bukharin, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adaptive budget allocation for parameter-efficient fine-tuning," *arXiv preprint arXiv:2303.10512*, 2023.

[138] N. Ding, X. Lv, Q. Wang, Y. Chen, B. Zhou, Z. Liu, and M. Sun, "Sparse low-rank adaptation of pre-trained language models," *arXiv preprint arXiv:2311.11696*, 2023.

[139] S. Haobo, H. Zhao, S. Majumder, and T. Lin, "Increasing model capacity for free: A simple strategy for parameter efficient fine-tuning," in *The Twelfth International Conference on Learning Representations*, 2023.

[140] R. Zhang, R. Qiang, S. A. Somayajula, and P. Xie, "Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning," *arXiv preprint arXiv:2403.09113*, 2024.

[141] A. X. Yang, M. Robeyns, X. Wang, and L. Aitchison, "Bayesian low-rank adaptation for large language models," *arXiv preprint arXiv:2308.13111*, 2023.

[142] Y. Lin, X. Ma, X. Chu, Y. Jin, Z. Yang, Y. Wang, and H. Mei, "Lora dropout as a sparsity regularizer for overfitting control," *arXiv preprint arXiv:2404.09610*, 2024.

[143] X. Meng, D. Dai, W. Luo, Z. Yang, S. Wu, X. Wang, P. Wang, Q. Dong, L. Chen, and Z. Sui, "Periodiclora: Breaking the low-rank bottleneck in lora optimization," *arXiv preprint arXiv:2402.16141*, 2024.

[144] S. Hayou, N. Ghosh, and B. Yu, "Lora+: Efficient low rank adaptation of large models," *arXiv preprint arXiv:2402.12354*, 2024.

[145] Y. Chen, S. Qian, H. Tang, X. Lai, Z. Liu, S. Han, and J. Jia, "Longlora: Efficient fine-tuning of long-context large language models," 2024. [Online]. Available: https://arxiv.org/abs/2309.12307

[146] C. Huang, Q. Liu, B. Y. Lin, T. Pang, C. Du, and M. Lin, "Lorahub: Efficient cross-task generalization via dynamic lora composition," *arXiv preprint arXiv:2307.13269*, 2023.

[147] Q. Liu, X. Wu, X. Zhao, Y. Zhu, D. Xu, F. Tian, and Y. Zheng, "Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications," *arXiv preprint arXiv:2310.18339*, 2023.

[148] W. Feng, C. Hao, Y. Zhang, Y. Han, and H. Wang, "Mixture-of-loras: An efficient multitask tuning for large language models," *arXiv preprint arXiv:2403.03432*, 2024.

[149] X. Wu, S. Huang, and F. Wei, "Mixture of lora experts," *arXiv preprint arXiv:2404.13628*, 2024.

[150] D. Li, Y. Ma, N. Wang, Z. Cheng, L. Duan, J. Zuo, C. Yang, and M. Tang, "Mixlora: Enhancing large language models fine-tuning with lora based mixture of experts," *arXiv preprint arXiv:2404.15159*, 2024.

[151] Y. Mao, L. Mathias, R. Hou, A. Almahairi, H. Ma, J. Han, W.-t. Yih, and M. Khabsa, "Unipelt: A unified framework for parameter-efficient language model tuning," *arXiv preprint arXiv:2110.07577*, 2021.

[152] J. Chen, A. Zhang, X. Shi, M. Li, A. Smola, and D. Yang, "Parameter-efficient fine-tuning design spaces," *arXiv preprint arXiv:2301.01821*, 2023.

[153] Y. Zhang, K. Zhou, and Z. Liu, "Neural prompt search," 2022.

[154] S. Zhong, J. Mo, and Z. Liu, "Autopet challenge 2022: Automatic segmentation of whole-body tumor lesion based on deep learning and fdg pet/ct," *arXiv preprint arXiv:2209.01212*, 2022.

[155] Z. Hu, Y. Lan, L. Wang, W. Xu, E.-P. Lim, R. K.-W. Lee, L. Bing, and S. Poria, "Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models," *arXiv preprint arXiv:2304.01933*, 2023.

[156] S. Hu, Z. Zhang, N. Ding, Y. Wang, Y. Wang, Z. Liu, and M. Sun, "Sparse structure search for parameter-efficient tuning," *arXiv preprint arXiv:2206.07382*, 2022.

[157] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685
[158] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *arXiv preprint arXiv:2304.08485*, 2023.
[159] P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan, "Learn to explain: Multimodal reasoning via thought chains for science question answering," 2022. [Online]. Available: https://arxiv.org/abs/2209.09513
[160] Y.-L. Sung, J. Cho, and M. Bansal, "Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks," 2022. [Online]. Available: https://arxiv.org/abs/2112.06825
[161] Y. Cui, W. Che, T. Liu, B. Qin, and Z. Yang, "Pre-training with whole word masking for chinese bert," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3504–3514, 2021.
[162] Y. Cui, W. Che, T. Liu, B. Qin, Z. Yang, S. Wang, and G. Hu, "Pre-training with whole word masking for chinese bert," *arXiv preprint arXiv:1906.08101*, 2019.
[163] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the association for computational linguistics*, vol. 8, pp. 64–77, 2020.
[164] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.
[165] 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," *arXiv preprint arXiv:1907.11692*, 2019.
[166] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.
[167] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.
[168] Y. Cui, W. Che, T. Liu, B. Qin, S. Wang, and G. Hu, "Revisiting pre-trained models for chinese natural language processing," *arXiv preprint arXiv:2004.13922*, 2020.
[169] H. Bao, L. Dong, S. Piao, and F. Wei, "Beit: Bert pre-training of image transformers," *arXiv preprint arXiv:2106.08254*, 2021.
[170] Z. Peng, L. Dong, H. Bao, Q. Ye, and F. Wei, "Beit v2: Masked image modeling with vector-quantized visual tokenizers," *arXiv preprint arXiv:2208.06366*, 2022.
[171] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som *et al.*, "Image as a foreign language: Beit pretraining for all vision and vision-language tasks," *arXiv preprint arXiv:2208.10442*, 2022.
[172] A. Vahdat, E. Andriyash, and W. Macready, "Dvae#: Discrete variational autoencoders with relaxed boltzmann priors," *Advances in Neural Information Processing Systems*, vol. 31, 2018.
[173] Z. Xu, "Roberta-wwm-ext fine-tuning for chinese text classification," *arXiv preprint arXiv:2103.00492*, 2021.
[174] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, "Ernie 2.0: A continual pre-training framework for language understanding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 8968–8975.
[175] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, Y. Lu *et al.*, "Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," *arXiv preprint arXiv:2107.02137*, 2021.
[176] F. Yu, J. Tang, W. Yin, Y. Sun, H. Tian, H. Wu, and H. Wang, "Ernie-vil: Knowledge enhanced vision-language representations through scene graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 3208–3216.
[177] B. Shan, W. Yin, Y. Sun, H. Tian, H. Wu, and H. Wang, "Ernie-vil 2.0: Multi-view contrastive learning for image-text pre-training," *arXiv preprint arXiv:2209.15270*, 2022.
[178] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.
[179] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
[180] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv preprint arXiv:2006.03654*, 2020.
[181] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.
[182] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.
[183] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
[184] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
[185] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," *arXiv preprint arXiv:1806.08730*, 2018.
[186] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
[187] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.
[188] "commoncrawl," 2022. [Online]. Available: https://commoncrawl.org/
[189] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.
[190] OpenAI, "Chatgpt," 2022. [Online]. Available: https://openai.com/chatgpt
[191] ——, "Gpt-4 technical report," 2023.
[192] S. Black, L. Gao, P. Wang, C. Leahy, and S. R. Biderman, "Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow," 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:245758737
[193] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1857–1867.
[194] A. Komatsuzaki, "Gpt-j-6b: 6b jax-based transformer," Jun 2021. [Online]. Available: https://arankomatsuzaki.wordpress.com/2021/06/04/gpt-j/
[195] C. Huebner, "Mesh transformer jax," Feb 2023. [Online]. Available: https://www.eleuther.ai/artifacts/mtj
[196] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang *et al.*, "Gpt-neox-20b: An open-source autoregressive language model," *arXiv preprint arXiv:2204.06745*, 2022.
[197] Microsoft, "Deepspeed is a deep learning optimization library that makes distributed training and inference easy, efficient, and effective." 2022. [Online]. Available: https://github.com/microsoft/DeepSpeed
[198] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan, "Dialogpt: Large-scale generative pre-training for conversational response generation," *arXiv preprint arXiv:1911.00536*, 2019.
[199] Google, "Introducing pathways: A next-generation ai architecture," 2022. [Online]. Available: https://blog.google/technology/ai/introducing-pathways-next-generation-ai-architecture/
[200] ——, "Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance," 2022. [Online]. Available: https://blog.research.google/2022/04/pathways-language-model-palm-scaling-to.html
[201] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.
[202] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint arXiv:2104.09864*, 2021.
[203] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.
[204] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[205] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[206] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International journal of computer vision*, vol. 123, pp. 32–73, 2017.

[207] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin *et al.*, "Scaling vision transformers to 22 billion parameters," in *International Conference on Machine Learning*. PMLR, 2023, pp. 7480–7512.

[208] H. Mehta, A. Thakurta, A. Kurakin, and A. Cutkosky, "Large scale transfer learning for differentially private image classification," *arXiv preprint arXiv:2205.02973*, 2022.

[209] S. Huang, L. Dong, W. Wang, Y. Hao, S. Singhal, S. Ma, T. Lv, L. Cui, O. K. Mohammed, Q. Liu *et al.*, "Language is not all you need: Aligning perception with language models," *arXiv preprint arXiv:2302.14045*, 2023.

[210] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo *et al.*, "Solving quantitative reasoning problems with language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3843–3857, 2022.

[211] X. Chen, X. Wang, S. Changpinyo, A. Piergiovanni, P. Padlewski, D. Salz, S. Goodman, A. Grycner, B. Mustafa, L. Beyer *et al.*, "Pali: A jointly-scaled multilingual language-image model," *arXiv preprint arXiv:2209.06794*, 2022.

[212] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[213] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 104–12 113.

[214] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, and T. Unterthiner, "Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[215] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "Palm 2 technical report," *arXiv preprint arXiv:2305.10403*, 2023.

[216] K. Singhal, T. Tu, J. Gottweis, R. Sayres, E. Wulczyn, L. Hou, K. Clark, S. Pfohl, H. Cole-Lewis, D. Neal *et al.*, "Towards expert-level medical question answering with large language models," *arXiv preprint arXiv:2305.09617*, 2023.

[217] D. Jin, E. Pan, N. Oufattole, W.-H. Weng, H. Fang, and P. Szolovits, "What disease does this patient have? a large-scale open domain question answering dataset from medical exams," *Applied Sciences*, vol. 11, no. 14, p. 6421, 2021.

[218] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl *et al.*, "Large language models encode clinical knowledge," *arXiv preprint arXiv:2212.13138*, 2022.

[219] H. Wang, S. Ma, S. Huang, L. Dong, W. Wang, Z. Peng, Y. Wu, P. Bajaj, S. Singhal, A. Benhaim *et al.*, "Foundation transformers," *arXiv preprint arXiv:2210.06423*, 2022.

[220] NVIDIA, "Nvidia nemo_2023," Oct 2023. [Online]. Available: https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/main/nlp/megatron.html

[221] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[222] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro *et al.*, "Efficient large-scale language model training on gpu clusters using megatron-lm," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[223] V. A. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro, "Reducing activation recomputation in large transformer models," *Proceedings of Machine Learning and Systems*, vol. 5, 2023.

[224] "Turing-nlg: A 17-billion-parameter language model by microsoft," Feb 2020. [Online]. Available: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

[225] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–16.

[226] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti *et al.*, "Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model," *arXiv preprint arXiv:2201.11990*, 2022.

[227] H.-C. Shin, Y. Zhang, E. Bakhturina, R. Puri, M. Patwary, M. Shoeybi, and R. Mani, "Biomegatron: Larger biomedical domain language model," *arXiv preprint arXiv:2010.06060*, 2020.

[228] P. Xu, M. Patwary, M. Shoeybi, R. Puri, P. Fung, A. Anandkumar, and B. Catanzaro, "Megatron-cntrl: Controllable story generation with external knowledge using large-scale language models," *arXiv preprint arXiv:2010.00840*, 2020.

[229] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[230] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Alpaca: A strong, replicable instruction-following model," 2019.

[231] OpenAI, "Models - openai," Feb 2020. [Online]. Available: https://platform.openai.com/docs/models

[232] "Guanaco - generative universal assistant for natural-language adaptive context-aware omnilingual outputs," Feb 2020. [Online]. Available: https://guanaco-model.github.io/

[233] "Alpaca-lora," Feb 2020. [Online]. Available: https://github.com/tloen/alpaca-lora

[234] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E. Gonzalez, I. Stoica, and E. P. Xing, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," March 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/

[235] ShareGPT, "Sharegpt," Feb 2020. [Online]. Available: https://sharegpt.com/

[236] Databricks, "dolly," Feb 2020. [Online]. Available: https://github.com/databrickslabs/dolly

[237] M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin, "Free dolly: Introducing the world's first truly open instruction-tuned llm," 2023. [Online]. Available: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm

[238] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff *et al.*, "Pythia: A suite for analyzing large language models across training and scaling," in *International Conference on Machine Learning*. PMLR, 2023, pp. 2397–2430.

[239] X. Geng, A. Gudibande, H. Liu, E. Wallace, P. Abbeel, S. Levine, and D. Song, "Koala: A dialogue model for academic research," Apr 2023. [Online]. Available: https://bair.berkeley.edu/blog/2023/04/03/koala/

[240] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," 2023.

[241] C. Xu, D. Guo, N. Duan, and J. McAuley, "Baize: An open-source chat model with parameter-efficient tuning on self-chat data," *arXiv preprint arXiv:2304.01196*, 2023.

[242] L. L. Ziang Leng, Qiyuan Chen, "Luotuo: Chinese-alpaca-lora," March 2023. [Online]. Available: https://github.com/LC1332/Chinese-alpaca-lora

[243] Y. Cui, Z. Yang, and X. Yao, "Efficient and effective text encoding for chinese llama and alpaca," *arXiv preprint arXiv:2304.08177*, 2023. [Online]. Available: https://arxiv.org/abs/2304.08177

[244] R. Zhang, J. Han, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li, P. Gao, and Y. Qiao, "Llama-adapter: Efficient fine-tuning of language models with zero-init attention," *arXiv preprint arXiv:2303.16199*, 2023.

[245] P. Gao, J. Han, R. Zhang, Z. Lin, S. Geng, A. Zhou, W. Zhang, P. Lu, C. He, X. Yue *et al.*, "Llama-adapter v2: Parameter-efficient visual instruction model," *arXiv preprint arXiv:2304.15010*, 2023.

[246] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *arXiv preprint arXiv:2304.10592*, 2023.

[247] C. LLaVA, "Chinese llava," July 2023. [Online]. Available: https://github.com/LinkSoul-AI/Chinese-LLaVA

[248] Y. Shu, S. Dong, G. Chen, W. Huang, R. Zhang, D. Shi, Q. Xiang, and Y. Shi, "Llasm: Large language and speech model," *arXiv preprint arXiv:2308.15930*, 2023.
[249] Visual-LLaMA, "Visual-llama," 2023. [Online]. Available: https://github.com/feizc/Visual-LLaMA
[250] H. Zhang, X. Li, and L. Bing, "Video-llama: An instruction-tuned audio-visual language model for video understanding," *arXiv preprint arXiv:2306.02858*, 2023.
[251] Q. Zhang, J. Zhang, Y. Xu, and D. Tao, "Vision transformer with quadrangle attention," *arXiv preprint arXiv:2303.15105*, 2023.
[252] K. Li, Y. He, Y. Wang, Y. Li, W. Wang, P. Luo, Y. Wang, L. Wang, and Y. Qiao, "Videochat: Chat-centric video understanding," *arXiv preprint arXiv:2305.06355*, 2023.
[253] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, "Imagebind: One embedding space to bind them all," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 15 180–15 190.
[254] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.
[255] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.
[256] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax
[257] T. Hennigan, T. Cai, T. Norman, L. Martens, and I. Babuschkin, "Haiku: Sonnet for JAX," 2020. [Online]. Available: http://github.com/deepmind/dm-haiku
[258] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.
[259] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 716–23 736, 2022.
[260] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1059–1071.
[261] A. Lab, "Ai21 studio logo," August 2021. [Online]. Available: https://www.ai21.com/studio
[262] O. Lieber, O. Sharir, B. Lenz, and Y. Shoham, "Jurassic-1: Technical details and evaluation," *White Paper. AI21 Labs*, vol. 1, 2021.
[263] E. Karpas, O. Abend, Y. Belinkov, B. Lenz, O. Lieber, N. Ratner, Y. Shoham, H. Bata, Y. Levine, K. Leyton-Brown *et al.*, "Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning," *arXiv preprint arXiv:2205.00445*, 2022.
[264] A. Lab, "Announcing jurassic-2 and task-specific apis," March 2022. [Online]. Available: https://www.ai21.com/blog/introducing-j2
[265] Anthropic, "Introducing claude," Mar 2023. [Online]. Available: https://www.anthropic.com/index/introducing-claude
[266] ——, "Claude 2," Jul 2023. [Online]. Available: https://www.anthropic.com/index/claude-2
[267] L. Mei, J. Mao, Z. Wang, C. Gan, and J. B. Tenenbaum, "Falcon: fast visual concept learning by integrating images, linguistic descriptions, and conceptual relations," *arXiv preprint arXiv:2203.16639*, 2022.
[268] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay, "The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only," *arXiv preprint arXiv:2306.01116*, 2023.
[269] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8821–8831.
[270] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.
[271] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.
[272] OpenAI, "Dall·e 3," September 2023. [Online]. Available: https://openai.com/dall-e-3
[273] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *International Conference on Machine Learning*. PMLR, 2023, pp. 28 492–28 518.
[274] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.
[275] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim *et al.*, "Starcoder: may the source be with you!" *arXiv preprint arXiv:2305.06161*, 2023.
[276] D. So, Q. Le, and C. Liang, "The evolved transformer," in *International conference on machine learning*. PMLR, 2019, pp. 5877–5886.
[277] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du *et al.*, "Lamda: Language models for dialog applications," *arXiv preprint arXiv:2201.08239*, 2022.
[278] G. H. Cohen, "Align: a program to superimpose protein coordinates, accounting for insertions and deletions," *Journal of applied crystallography*, vol. 30, no. 6, pp. 1160–1161, 1997.
[279] B. Koonce and B. Koonce, "Efficientnet," *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, pp. 109–123, 2021.
[280] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat *et al.*, "Glam: Efficient scaling of language models with mixture-of-experts," in *International Conference on Machine Learning*. PMLR, 2022, pp. 5547–5569.
[281] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.
[282] "Phi-1.5," 2023. [Online]. Available: https://huggingface.co/microsoft/phi-1_5
[283] "Phi-2: The surprising power of small language models," 2023. [Online]. Available: https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/
[284] C. Li, H. Xu, J. Tian, W. Wang, M. Yan, B. Bi, J. Ye, H. Chen, G. Xu, Z. Cao *et al.*, "mplug: Effective and efficient vision-language learning by cross-modal skip-connections," *arXiv preprint arXiv:2205.12005*, 2022.
[285] H. Xu, Q. Ye, M. Yan, Y. Shi, J. Ye, Y. Xu, C. Li, B. Bi, Q. Qian, W. Wang *et al.*, "mplug-2: A modularized multi-modal foundation model across text, image and video," *arXiv preprint arXiv:2302.00402*, 2023.
[286] Z. Yang, L. Li, J. Wang, K. Lin, E. Azarnasab, F. Ahmed, Z. Liu, C. Liu, M. Zeng, and L. Wang, "Mm-react: Prompting chatgpt for multimodal reasoning and action," *arXiv preprint arXiv:2303.11381*, 2023.
[287] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface," *arXiv preprint arXiv:2303.17580*, 2023.
[288] H. Xu, Q. Ye, X. Wu, M. Yan, Y. Miao, J. Ye, G. Xu, A. Hu, Y. Shi, G. Xu *et al.*, "Youku-mplug: A 10 million large-scale chinese video-language dataset for pre-training and benchmarks," *arXiv preprint arXiv:2306.04362*, 2023.
[289] J. Ye, A. Hu, H. Xu, Q. Ye, M. Yan, Y. Dan, C. Zhao, G. Xu, C. Li, J. Tian *et al.*, "mplug-docowl: Modularized multimodal large language model for document understanding," *arXiv preprint arXiv:2307.02499*, 2023.
[290] K. Sanders, D. Etter, R. Kriz, and B. Van Durme, "Multivent: Multilingual videos of events with aligned natural text," *arXiv preprint arXiv:2307.03153*, 2023.
[291] S. Soltan, S. Ananthakrishnan, J. FitzGerald, R. Gupta, W. Hamza, H. Khan, C. Peris, S. Rawls, A. Rosenbaum, A. Rumshisky *et al.*, "Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model," *arXiv preprint arXiv:2208.01448*, 2022.
[292] S. Bao, H. He, F. Wang, H. Wu, and H. Wang, "Plato: Pre-trained dialogue generation model with discrete latent variable," *arXiv preprint arXiv:1910.07931*, 2019.
[293] S. Bao, H. He, F. Wang, H. Wu, H. Wang, W. Wu, Z. Guo, Z. Liu, and X. Xu, "Plato-2: Towards building an open-domain chatbot via curriculum learning," *arXiv preprint arXiv:2006.16779*, 2020.
[294] BAAI, "Baai 23," 2023. [Online]. Available: https://2023.baai.ac.cn/about
[295] Y. Huo, M. Zhang, G. Liu, H. Lu, Y. Gao, G. Yang, J. Wen, H. Zhang, B. Xu, W. Zheng *et al.*, "Wenlan: Bridging vision and language by large-scale multi-modal pre-training," *arXiv preprint arXiv:2103.06561*, 2021.
[296] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, "Learning deep structured semantic models for web search using clickthrough

data," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 2333–2338.

[297] M. Ding, Z. Yang, W. Hong, W. Zheng, C. Zhou, D. Yin, J. Lin, X. Zou, Z. Shao, H. Yang *et al.*, "Cogview: Mastering text-to-image generation via transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 822–19 835, 2021.

[298] C. Xiao, X. Hu, Z. Liu, C. Tu, and M. Sun, "Lawformer: A pre-trained language model for chinese legal long documents," *AI Open*, vol. 2, pp. 79–84, 2021.

[299] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[300] S. Iyer, X. V. Lin, R. Pasunuru, T. Mihaylov, D. Simig, P. Yu, K. Shuster, T. Wang, Q. Liu, P. S. Koura *et al.*, "Opt-iml: Scaling language model instruction meta learning through the lens of generalization," *arXiv preprint arXiv:2212.12017*, 2022.

[301] M. Khrushchev, R. Vasilev, A. Petrov, and N. Zinov, "YaLM 100B," Jun. 2022. [Online]. Available: https://github.com/yandex/YaLM-100B

[302] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé *et al.*, "Bloom: A 176b-parameter open-access multilingual language model," *arXiv preprint arXiv:2211.05100*, 2022.

[303] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," *arXiv preprint arXiv:2211.09085*, 2022.

[304] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[305] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[306] H. A. Chipman, E. I. George, R. E. McCulloch, and T. S. Shively, "mbart: multidimensional monotone bart," *Bayesian Analysis*, vol. 17, pp. 515–544, 2022.

[307] S. Zhang, V. Chaudhary, N. Goyal, J. Cross, G. Wenzek, M. Bansal, and F. Guzman, "How robust is neural machine translation to language imbalance in multilingual tokenizer training?" *arXiv preprint arXiv:2204.14268*, 2022.

[308] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.

[309] F. Christopoulou, G. Lampouras, M. Gritta, G. Zhang, Y. Guo, Z. Li, Q. Zhang, M. Xiao, B. Shen, L. Li *et al.*, "Pangu-coder: Program synthesis with function-level language modeling," *arXiv preprint arXiv:2207.11280*, 2022.

[310] S. Yuan, H. Zhao, Z. Du, M. Ding, X. Liu, Y. Cen, X. Zou, Z. Yang, and J. Tang, "Wudaocorpora: A super large-scale chinese corpora for pre-training language models," *AI Open*, vol. 2, pp. 65–68, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000152

[311] L. Xu, X. Zhang, and Q. Dong, "Cluecorpus2020: A large-scale chinese corpus for pre-training language model," *arXiv preprint arXiv:2003.01355*, 2020.

[312] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[313] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 5232–5270, 2022.

[314] "superglue benchmarking," 2022. [Online]. Available: https://super.gluebenchmark.com/

[315] T. Schick and H. Schütze, "Exploiting cloze questions for few shot text classification and natural language inference," *arXiv preprint arXiv:2001.07676*, 2020.

[316] "Glm6b," 2022. [Online]. Available: https://github.com/THUDM/ChatGLM-6B/blob/main/README_en.md

[317] "Chinese and english multimodal conversational langauge model," 2022. [Online]. Available: https://github.com/THUDM/VisualGLM-6B

[318] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023.

[319] C. Jiang, H. Xu, W. Ye, Q. Ye, C. Li, M. Yan, B. Bi, S. Zhang, J. Zhang, and F. Huang, "Copa: Efficient vision-language pre-training through collaborative object- and patch-text alignment," 2024. [Online]. Available: https://arxiv.org/abs/2308.03475

[320] J. Liu, X. Huang, J. Zheng, Y. Liu, and H. Li, "Mixmae: Mixed and masked autoencoder for efficient pretraining of hierarchical vision transformers," 2023. [Online]. Available: https://arxiv.org/abs/2205.13137

[321] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini, "Deduplicating training data makes language models better," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8424–8445.

[322] C. Jiang, H. Xu, C. Li, M. Yan, W. Ye, S. Zhang, B. Bi, and S. Huang, "TRIPS: Efficient vision-and-language pre-training with text-relevant image patch selection," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 4084–4096.

[323] Y. Liu, C. Matsoukas, F. Strand, H. Azizpour, and K. Smith, "Patchdropout: Economizing vision transformers using patch dropout," 2022. [Online]. Available: https://arxiv.org/abs/2208.07220

[324] S. Wei, T. Ye, S. Zhang, Y. Tang, and J. Liang, "Joint token pruning and squeezing towards more aggressive compression of vision transformers," 2023. [Online]. Available: https://arxiv.org/abs/2304.10716

[325] H. Wang, C. Ge, H. Chen, and X. Sun, "Prenas: Preferred one-shot learning towards efficient neural architecture search," 2023. [Online]. Available: https://arxiv.org/abs/2304.14636

[326] O. Bohdal, L. Balles, M. Wistuba, B. Ermis, C. Archambeau, and G. Zappella, "Pasha: Efficient hpo and nas with progressive resource allocation," 2023. [Online]. Available: https://arxiv.org/abs/2207.06940

[327] G. Li, Y. Yang, K. Bhardwaj, and R. Marculescu, "Zico: Zero-shot nas via inverse coefficient of variation on gradients," 2023. [Online]. Available: https://arxiv.org/abs/2301.11300

[328] C. Tang, L. L. Zhang, H. Jiang, J. Xu, T. Cao, Q. Zhang, Y. Yang, Z. Wang, and M. Yang, "Elasticvit: Conflict-aware supernet training for deploying fast vision transformer on diverse mobile devices," 2023. [Online]. Available: https://arxiv.org/abs/2303.09730

[329] C. Hu, C. Wang, X. Ma, X. Meng, Y. Li, T. Xiao, J. Zhu, and C. Li, "RankNAS: Efficient neural architecture search by pairwise ranking," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2469–2480.

[330] P. Wang, R. Panda, L. T. Hennigen, P. Greengard, L. Karlinsky, R. Feris, D. D. Cox, Z. Wang, and Y. Kim, "Learning to grow pretrained models for efficient transformer training," 2023. [Online]. Available: https://arxiv.org/abs/2303.00980

[331] S. Shen, P. Walsh, K. Keutzer, J. Dodge, M. Peters, and I. Beltagy, "Staged training for transformer language models," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 19 893–19 908. [Online]. Available: https://proceedings.mlr.press/v162/shen22f.html

[332] Y. Qin, Y. Lin, J. Yi, J. Zhang, X. Han, Z. Zhang, Y. Su, Z. Liu, P. Li, M. Sun, and J. Zhou, "Knowledge inheritance for pre-trained language models," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 3921–3937. [Online]. Available: https://aclanthology.org/2022.naacl-main.288

[333] X. Gu, L. Liu, H. Yu, J. Li, C. Chen, and J. Han, "On the transformer growth for progressive BERT training," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds. Online: Association for Computational Linguistics, Jun. 2021, pp. 5174–5180. [Online]. Available: https://aclanthology.org/2021.naacl-main.406

[334] L. Gong, D. He, Z. Li, T. Qin, L. Wang, and T. Liu, "Efficient training of BERT by progressively stacking," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds.,

vol. 97. PMLR, 09–15 Jun 2019, pp. 2337–2346. [Online]. Available: https://proceedings.mlr.press/v97/gong19a.html
[335] Z. Pan, P. Chen, H. He, J. Liu, J. Cai, and B. Zhuang, "Mesa: A memory-saving training framework for transformers," 2022. [Online]. Available: https://arxiv.org/abs/2111.11124
[336] X. Liu, L. Zheng, D. Wang, Y. Cen, W. Chen, X. Han, J. Chen, Z. Liu, J. Tang, J. Gonzalez, M. Mahoney, and A. Cheung, "Gact: Activation compressed training for generic network architectures," 2022. [Online]. Available: https://arxiv.org/abs/2206.11357
[337] A. Chakrabarti and B. Moseley, "Backprop with approximate activations for memory-efficient network training," 2019. [Online]. Available: https://arxiv.org/abs/1901.07988
[338] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," 2018. [Online]. Available: https://arxiv.org/abs/1710.03740
[339] "New models and developer products announced at devday," 2023. [Online]. Available: https://openai.com/blog/new-models-and-developer-products-announced-at-devday
[340] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.
[341] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Fast model editing at scale," *arXiv preprint arXiv:2110.11309*, 2021.
[342] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn, "Memory-based model editing at scale," in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 817–15 831.
[343] M. Reid and G. Neubig, "Learning to model editing processes," *arXiv preprint arXiv:2205.12374*, 2022.
[344] Q. He, Z. Li, and X. Zhang, "Data deduplication techniques," in *2010 international conference on future information technology and management engineering*, vol. 1. IEEE, 2010, pp. 430–433.
[345] P. P. Liang, C. Wu, L.-P. Morency, and R. Salakhutdinov, "Towards understanding and mitigating social biases in language models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6565–6576.
[346] S. Bordia and S. R. Bowman, "Identifying and reducing gender bias in word-level language models," *arXiv preprint arXiv:1904.03035*, 2019.
[347] R. Liu, C. Jia, J. Wei, G. Xu, L. Wang, and S. Vosoughi, "Mitigating political bias in language models through reinforced calibration," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 14 857–14 866.
[348] M. Nadeem, A. Bethke, and S. Reddy, "Stereoset: Measuring stereotypical bias in pretrained language models," *arXiv preprint arXiv:2004.09456*, 2020.
[349] V. Sanh, T. Wolf, and A. Rush, "Movement pruning: Adaptive sparsity by fine-tuning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 378–20 389, 2020.
[350] R. Cheong and R. Daniel, "transformers. zip: Compressing transformers with pruning and quantization," *Technical report, tech. rep., Stanford University, Stanford, California*, 2019.
[351] M. A. Gordon, K. Duh, and N. Andrews, "Compressing bert: Studying the effects of weight pruning on transfer learning," *arXiv preprint arXiv:2002.08307*, 2020.
[352] Z. Wang, J. Wohlwend, and T. Lei, "Structured pruning of large language models," *arXiv preprint arXiv:1910.04732*, 2019.
[353] B. Cui, Y. Li, and Z. Zhang, "Joint structured pruning and dense knowledge distillation for efficient transformer model compression," *Neurocomputing*, vol. 458, pp. 56–69, 2021.
[354] Z. Yang, Y. Cui, and Z. Chen, "Textpruner: A model pruning toolkit for pre-trained language models," *arXiv preprint arXiv:2203.15996*, 2022.
[355] E. Frantar and D. Alistarh, "Massive language models can be accurately pruned in one-shot," *arXiv preprint arXiv:2301.00774*, 2023.
[356] M. Zhang, C. Shen, Z. Yang, L. Ou, X. Yu, B. Zhuang *et al.*, "Pruning meets low-rank parameter-efficient fine-tuning," *arXiv preprint arXiv:2305.18403*, 2023.
[357] T. Chen, T. Ding, B. Yadav, I. Zharkov, and L. Liang, "Lorashear: Efficient large language model structured pruning and knowledge recovery," *arXiv preprint arXiv:2310.18356*, 2023.
[358] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," *arXiv preprint arXiv:2306.11695*, 2023.
[359] M. Xia, Z. Zhong, and D. Chen, "Structured pruning learns compact and accurate models," *arXiv preprint arXiv:2204.00408*, 2022.
[360] M. Zhu, Y. Tang, and K. Han, "Vision transformer pruning," *arXiv preprint arXiv:2104.08500*, 2021.
[361] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418*, 2019.
[362] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," *arXiv preprint arXiv:1909.11556*, 2019.
[363] F. Lagunas, E. Charlaix, V. Sanh, and A. M. Rush, "Block pruning for faster transformers," *arXiv preprint arXiv:2109.04838*, 2021.
[364] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" *Advances in neural information processing systems*, vol. 32, 2019.
[365] D. Campos, A. Marques, T. Nguyen, M. Kurtz, and C. Zhai, "Sparse* bert: Sparse models are robust," *arXiv preprint arXiv:2205.12452*, 2022.
[366] M. Santacroce, Z. Wen, Y. Shen, and Y. Li, "What matters in the structured pruning of generative language models?" *arXiv preprint arXiv:2302.03773*, 2023.
[367] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," *arXiv preprint arXiv:2305.11627*, 2023.
[368] S. Guo, J. Xu, L. L. Zhang, and M. Yang, "Compresso: Structured pruning with collaborative prompting learns compact large language models," *arXiv preprint arXiv:2310.05015*, 2023.
[369] M. Xia, T. Gao, Z. Zeng, and D. Chen, "Sheared llama: Accelerating language model pre-training via structured pruning," *arXiv preprint arXiv:2310.06694*, 2023.
[370] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu, and P. Micikevicius, "Accelerating sparse deep neural networks," *arXiv preprint arXiv:2104.08378*, 2021.
[371] C. Fang, A. Zhou, and Z. Wang, "An algorithm–hardware co-optimized framework for accelerating n: M sparse transformers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1573–1586, 2022.
[372] C. Holmes, M. Zhang, Y. He, and B. Wu, "Nxmtransformer: Semi-structured sparsification for natural language understanding via admm," *Advances in neural information processing systems*, vol. 34, pp. 1818–1830, 2021.
[373] C. Fang, S. Guo, W. Wu, J. Lin, Z. Wang, M. K. Hsu, and L. Liu, "An efficient hardware accelerator for sparse transformer neural networks," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2022, pp. 2670–2674.
[374] R. Xu, F. Luo, C. Wang, B. Chang, J. Huang, S. Huang, and F. Huang, "From dense to sparse: Contrastive pruning for better pre-trained language model compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 11 547–11 555.
[375] J. Zhang, Y. Zhou, and R. Saab, "Post-training quantization for neural networks with provable guarantees," *SIAM Journal on Mathematics of Data Science*, vol. 5, no. 2, pp. 373–399, 2023.
[376] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.
[377] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, "Llm-qat: Data-free quantization aware training for large language models," *arXiv preprint arXiv:2305.17888*, 2023.
[378] J. Kim, J. H. Lee, S. Kim, J. Park, K. M. Yoo, S. J. Kwon, and D. Lee, "Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization," *arXiv preprint arXiv:2305.14152*, 2023.
[379] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *arXiv preprint arXiv:2305.14314*, 2023.
[380] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.
[381] ——, "Optq: Accurate quantization for generative pre-trained transformers," in *The Eleventh International Conference on Learning Representations*, 2022.
[382] Z. Yuan, L. Niu, J. Liu, W. Liu, X. Wang, Y. Shang, G. Sun, Q. Wu, J. Wu, and B. Wu, "Rptq: Reorder-based post-training quantization for large language models," *arXiv preprint arXiv:2304.01089*, 2023.
[383] Q. Li, Y. Zhang, L. Li, P. Yao, B. Zhang, X. Chu, Y. Sun, L. Du, and Y. Xie, "Fptq: Fine-grained post-training quantization for large language models," *arXiv preprint arXiv:2308.15987*, 2023.
[384] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "Zeroquant: Efficient and affordable post-training quantization for

large-scale transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 168–27 183, 2022.
[385] Z. Yao, X. Wu, C. Li, S. Youn, and Y. He, "Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation," *arXiv preprint arXiv:2303.08302*, 2023.
[386] X. Wu, Z. Yao, and Y. He, "Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats," *arXiv preprint arXiv:2307.09782*, 2023.
[387] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 087–38 099.
[388] W. Shao, M. Chen, Z. Zhang, P. Xu, L. Zhao, Z. Li, K. Zhang, P. Gao, Y. Qiao, and P. Luo, "Omniquant: Omnidirectionally calibrated quantization for large language models," *arXiv preprint arXiv:2308.13137*, 2023.
[389] C. Lee, J. Jin, T. Kim, H. Kim, and E. Park, "Owq: Lessons learned from activation outliers for weight quantization in large language models," *arXiv preprint arXiv:2306.02272*, 2023.
[390] J. Lin, J. Tang, H. Tang, S. Yang, X. Dang, and S. Han, "Awq: Activation-aware weight quantization for llm compression and acceleration," *arXiv preprint arXiv:2306.00978*, 2023.
[391] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm. int8 (): 8-bit matrix multiplication for transformers at scale," *arXiv preprint arXiv:2208.07339*, 2022.
[392] X. Wu, C. Li, R. Y. Aminabadi, Z. Yao, and Y. He, "Understanding int4 quantization for language models: Latency speedup, composability, and failure cases," *ICML'23: Proceedings of the 40th International Conference on Machine Learning*, 2023.
[393] D. Abati, H. B. Yahia, M. Nagel, and A. Habibian, "Resq: Residual quantization for video perception-supplementary material," *ICCV2023*, 2023.
[394] S. Kim, C. Hooper, A. Gholami, Z. Dong, X. Li, S. Shen, M. W. Mahoney, and K. Keutzer, "Squeezellm: Dense-and-sparse quantization," *arXiv preprint arXiv:2306.07629*, 2023.
[395] J. Chee, Y. Cai, V. Kuleshov, and C. De Sa, "Quip: 2-bit quantization of large language models with guarantees," *arXiv preprint arXiv:2307.13304*, 2023.
[396] W. Cheng, W. Zhang, H. Shen, Y. Cai, X. He, and K. Lv, "Optimize weight rounding via signed gradient descent for the quantization of llms," *arXiv preprint arXiv:2309.05516*, 2023.
[397] L. Li, Q. Li, B. Zhang, and X. Chu, "Norm tweaking: High-performance low-bit quantization of large language models," *arXiv preprint arXiv:2309.02784*, 2023.
[398] C. Guo, J. Tang, W. Hu, J. Leng, C. Zhang, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "Olive: Accelerating large language models via hardware-friendly outlier-victim pair quantization," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–15.
[399] K. Behdin, A. Acharya, A. Gupta, S. Keerthi, and R. Mazumder, "Quantease: Optimization-based quantization for language models–an efficient and intuitive algorithm," *arXiv preprint arXiv:2309.01885*, 2023.
[400] X. Wei, Y. Zhang, X. Zhang, R. Gong, S. Zhang, Q. Zhang, F. Yu, and X. Liu, "Outlier suppression: Pushing the limit of low-bit transformer language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 402–17 414, 2022.
[401] X. Wei, Y. Zhang, Y. Li, X. Zhang, R. Gong, J. Guo, and X. Liu, "Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling," *arXiv preprint arXiv:2304.09145*, 2023.
[402] G. Park, B. Park, M. Kim, S. Lee, J. Kim, B. Kwon *et al.*, "Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models," 2022.
[403] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.
[404] C. Alberti, J. Ling, M. Collins, and D. Reitter, "Fusion of detected objects in text for visual question answering," *arXiv preprint arXiv:1908.05054*, 2019.
[405] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," *arXiv preprint arXiv:1908.07490*, 2019.
[406] J. Cho, J. Lu, D. Schwenk, H. Hajishirzi, and A. Kembhavi, "X-lxmert: Paint, caption and answer questions with multi-modal transformers," *arXiv preprint arXiv:2009.11278*, 2020.
[407] Y.-C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "Uniter: Universal image-text representation learning," in *European conference on computer vision*. Springer, 2020, pp. 104–120.
[408] G. Li, N. Duan, Y. Fang, M. Gong, and D. Jiang, "Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 11 336–11 344.
[409] Z. Huang, Z. Zeng, B. Liu, D. Fu, and J. Fu, "Pixel-bert: Aligning image pixels with text by deep multi-modal transformers," *arXiv preprint arXiv:2004.00849*, 2020.
[410] W. Li, C. Gao, G. Niu, X. Xiao, H. Liu, J. Liu, H. Wu, and H. Wang, "Unimo: Towards unified-modal understanding and generation via cross-modal contrastive learning," *arXiv preprint arXiv:2012.15409*, 2020.
[411] ——, "Unimo-2: End-to-end unified vision-language grounded learning," *arXiv preprint arXiv:2203.09067*, 2022.
[412] S. Yuan, Z. Shuai, L. Jiahong, X. Zhao, Z. Hanyu, and T. Jie, "Wudaomm: A large-scale multi-modal dataset for pre-training models," *arXiv preprint arXiv:2203.11480*, 2022.
[413] Y. Liu, G. Zhu, B. Zhu, Q. Song, G. Ge, H. Chen, G. Qiao, R. Peng, L. Wu, and J. Wang, "Taisu: A 166m large-scale high-quality dataset for chinese vision-language pre-training," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 705–16 717, 2022.
[414] Y. Zeng, C. Jiang, J. Mao, J. Han, C. Ye, Q. Huang *et al.*, "Clip2: Contrastive language-image-point pretraining from real-world point cloud data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 15 244–15 253.
[415] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5583–5594.
[416] J. Lin, A. Yang, Y. Zhang, J. Liu, J. Zhou, and H. Yang, "Interbert: Vision-and-language interaction for multi-modal pretraining," *arXiv preprint arXiv:2003.13198*, 2020.
[417] D. Qi, L. Su, J. Song, E. Cui, T. Bharti, and A. Sacheti, "Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data," *arXiv preprint arXiv:2001.07966*, 2020.
[418] T. L. Scao and A. M. Rush, "How many data points is a prompt worth?" *arXiv preprint arXiv:2103.08493*, 2021.
[419] S. Bao, H. He, F. Wang, H. Wu, H. Wang, W. Wu, Z. Wu, Z. Guo, H. Lu, X. Huang *et al.*, "Plato-xl: Exploring the large-scale pre-training of dialogue generation," *arXiv preprint arXiv:2109.09519*, 2021.
[420] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "Bloomberggpt: A large language model for finance," *arXiv preprint arXiv:2303.17564*, 2023.
[421] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "Ctrl: A conditional transformer language model for controllable generation," *arXiv preprint arXiv:1909.05858*, 2019.
[422] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "Coca: Contrastive captioners are image-text foundation models," *arXiv preprint arXiv:2205.01917*, 2022.
[423] J. Li, R. Selvaraju, A. Gotmare, S. Joty, C. Xiong, and S. C. H. Hoi, "Align before fuse: Vision and language representation learning with momentum distillation," *Advances in neural information processing systems*, vol. 34, pp. 9694–9705, 2021.
[424] Z. Wang, J. Yu, A. W. Yu, Z. Dai, Y. Tsvetkov, and Y. Cao, "Simvlm: Simple visual language model pretraining with weak supervision," *arXiv preprint arXiv:2108.10904*, 2021.
[425] H. Bao, W. Wang, L. Dong, Q. Liu, O. K. Mohammed, K. Aggarwal, S. Som, S. Piao, and F. Wei, "Vlmo: Unified vision-language pre-training with mixture-of-modality-experts," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 897–32 912, 2022.
[426] Z. Huang, Z. Zeng, Y. Huang, B. Liu, D. Fu, and J. Fu, "Seeing out of the box: End-to-end pre-training for vision-language representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 976–12 985.
[427] J. Lin, Y. Qu, W. Guo, X. Dai, R. Tang, Y. Yu, and W. Zhang, "Map: A model-agnostic pretraining framework for click-through rate prediction," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 1384–1395.

A. MODEL PARAMETERS

As the evolution of LLMs, the number of parameters is also evolving from the BERT and GPT with only millions level of data to current trillion level of model parameters, the table below shows the models architecture and other metrics.

TABLE 12. LLMs categorized into their Architectures, parameter count, and the Base LLM from which they are derived.

Model	Architecture	Parameter	Base
BERT	Auto-encoding	110M, 340M	-
SpanBERT	Auto-encoding	110M, 340M	BERT
RoBERTa	Auto-encoding	125M, 355M	BERT
DistilBERT	Auto-encoding	66M	BERT
BEiT	Auto-encoding	86M, 307M	BERT
Transformer-XL	Auto-encoding	257M	-
XLNet	Auto-encoding	110M, 240M	Transformer-XL
ERNIE	Auto-encoding	110M	BERT
ERNIE 2.0	Auto-encoding	110M	ERNIE
ERNIE 3.0	Auto-encoding	260B	ERNIE 2.0
ALBERT	Auto-encoding	125M	BERT
ELECTRA	Auto-encoding	14M, 110M, 335M	BERT
DeBERTA	Auto-encoding	100M, 350M, 700M	BERT
BART	Sequence to Sequence	140M, 400M	-
T5	Sequence to Sequence	60M, 220M, 770M	-
FLAN	Sequence to Sequence	60M, 250M, 780M, 3B, 11B	T5
Pangu-alpha	Sequence to Sequence	2.6B, 13B, 200B	-
Pangu-sigma	Sequence to Sequence	1.085T	Pangu-alpha
GLM	Sequence to Sequence	130B	-
Minerva	Sequence to Sequence	540B	-
GPT	Auto-regressive	117M	-
GPT-2	Auto-regressive	1.5B	GPT
GPT-3	Auto-regressive	175B	GPT-2
GPT-Neo	Auto-regressive	125M, 1.3B, 2.7B	GPT-3
GPT-J	Auto-regressive	6B	GPT-3
GPT-NeoX	Auto-regressive	20B	GPT-Neo
PaLM	Auto-regressive	540B	-
PaLM-E	Auto-regressive	562B	PaLM, ViT
PaLi	Auto-regressive	3B, 15B, 17B	PaLM
PaLM-2	Auto-regressive	340B	PaLM
KOSMOS-1	Auto-regressive	1.6B	-
Megatron LM	Auto-regressive	1.2B, 2.5B, 4.2B, 8.3B	-
Turing NLG	Auto-regressive	17B	-
Megatron-Turing NLG	Auto-regressive	530B	Megatron LM, Turing NLG
LLaMA	Auto-regressive	7B, 13B, 33B, 65B	-
Alpaca	Auto-regressive	7B, 13B, 30B, 65B	LLaMA
Guanaco	Auto-regressive	7B, 13B, 30B, 65B	LLaMA
Vicuna	Auto-regressive	7B, 13B, 30B, 65B	LLaMA
Dolly	Auto-regressive	6B	LLaMA
Dolly v2	Auto-regressive	12B	Dolly
Pythia	Auto-regressive	70M, 160M, 410M, 1B, 1.4B	-
FastChat	Auto-regressive	3B	-
LLaMA 2	Auto-regressive	7B, 13B, 34B, 70B	LLaMA
Baize	Auto-regressive	7B, 13B	LLaMA
LLaVA	Auto-regressive	13B	-
Gopher	Auto-regressive	44M, 117M, 417M, 1.4B, 7.1B	-
Chinchilla	Auto-regressive	70B, 280B	Gopher
Flamingo	Auto-regressive	80B	Chinchilla
Jurassic-1	Auto-regressive	7.5B, 178B	-
Claude	Auto-regressive	52B	-
Claude 2	Auto-regressive	130B	Claude
Falcon	Auto-regressive	40B, 180B	-

DALL-E	Auto-regressive	12B	GPT-3
DALLE-E 2	Auto-regressive	3.5B	CLIP
Whisper	Auto-regressive	74M, 244M, 769M, 1550M	-
Codex	Auto-regressive	12B	-
LaMDA	Auto-regressive	137B	-
GaLM	Auto-regressive	1.2T	-
mPLUG	Auto-regressive	14M	-
mPLUG-Owl	Auto-regressive	7B	mPLUG, ViT
AlexaTM	Auto-regressive	20B	-
PLATO-2	Auto-regressive	1.6B	PLATO
PLATO-XL [419]	Auto-regressive	11B	PLATO-2
OPT	Auto-regressive	175B	-
YaLM	Auto-regressive	100B	-
BLOOM	Auto-regressive	176B	-
Galactica	Auto-regressive	120B	-
VILBERT	Auto-regressive	3B	BERT
UNITER [407]	Auto-regressive	303M	-
Unicoder-VL	Auto-regressive	195M	BERT
ERNIE-VILG	Auto-regressive	10B	ERNIE
ERNIE-VIL 2.0	Auto-regressive	24B	ERNIE-VILG
CLIP	Auto-regressive	63M	-
ViLT	Auto-regressive	87M	-
BloombergGPT [420]	Auto-regressive	50B	-
CTRL [421]	Auto-regressive	1.6B	-

B. MULTIMODAL SUPPORT

The table below is the multimodal support for each of the multimodal LLMs mentioned in this survey, the multimodality was classified into the following categories: Text, Image, Video, Audio, Embodied

TABLE 13. Multimodal support for MLLMs

Model	Text	Image	Video	Audio	Embodied	Model	Text	Image	Video	Audio	Embodied
VisualBERT	✓	✓	-	-	-	mPLUG-owl	✓	✓	✓	-	-
BEiT	✓	✓	-	-	-	mPLUG-DOCOWL	✓	✓	-	-	-
BEiT v2	✓	✓	-	-	-	WenLan	✓	✓	-	-	-
BEiT v3	✓	✓	-	-	-	VILBERT	✓	✓	-	-	-
ERNIE-VilG	✓	✓	-	-	-	B2T2	✓	✓	-	-	-
ERNIE-Vil 2.0	✓	✓	-	-	-	LXMERT	✓	✓	-	-	-
VisualGLM	✓	✓	-	-	-	XLXMERT	✓	✓	-	-	-
GPT-4	✓	✓	-	-	-	UNITER	✓	✓	-	-	-
PaLM-E	✓	✓	✓	✓	✓	Unicoder-VL	✓	✓	-	-	-
KOSMOS-1	✓	✓	✓	✓	-	Pixel-BERT	✓	✓	-	-	-
PaLi	✓	✓	-	-	-	UNIMO	✓	✓	-	-	-
LLaMA adapter	✓	✓	-	-	-	UNIMO 2	✓	✓	-	-	-
LLaMA adapter v2	✓	✓	-	-	-	BLIP	✓	✓	-	-	-
MiniGPT-4	✓	✓	-	-	-	BLIP 2	✓	✓	-	-	-
LLaSM	✓	✓	✓	✓	-	BLIP 3	✓	✓	-	-	-
Video-LLaMA	✓	-	✓	-	-	WudaoMM	✓	✓	-	-	-
LLaVA	✓	✓	-	-	-	CLIP2	✓	✓	-	-	-
VideoChat	✓	-	✓	-	-	ViLT	✓	✓	-	-	-
Flamingo	✓	✓	✓	-	-	InterBERT	✓	✓	-	-	-
DALL-E	✓	✓	-	-	-	ImageBERT	✓	✓	-	-	-
DALL-E 2	✓	✓	-	-	-	Med-PaLM	✓	✓	-	✓	-
CLIP	✓	✓	-	-	-	Med-PaLM 2	✓	✓	-	✓	-
Whisper	✓	-	-	✓	-	OSCAR	✓	✓	-	✓	-
ALIGN	✓	✓	-	-	-	Virtex	✓	✓	-	-	-
mPLUG	✓	✓	✓	-	-	VILLA	✓	✓	-	✓	-
mPLUG 2	✓	✓	✓	-	-	BARD	✓	✓	-	✓	-
SLIP	✓	✓	-	-	-	FLIP	✓	✓	-	-	-

C. TRAINING APPROACHES OF MULTIMODAL LARGE LANGUAGE MODELS

The table below outlines the commonly used training approaches for multimodal large language models, highlighting techniques for several models discussed in this survey paper. It compares the data inputs, integration algorithms, and training objectives across different models.

TABLE 14. Multimodal training approaches for MLLMs

Model	LM	MLM	ITM	MRC	MRM	WRA	Seq2Seq	CMCL	VQA	MRFR	SGP	VLC	MTL	WPA	MVM	VLM	MSM	ITC	ITG	PrefixLM	MOC
VisualBERT	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
BEiT	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
BEiT v2	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
BEiT v3	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-
ERNIE-VilG	-	✓	✓	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-
ERNIE-Vil 2.0	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-
KOSMOS-1	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MiniGPT-4	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-
LLaVA	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
Flamingo	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-
CLIP	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
Whisper	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-
ALIGN	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
mPLUG	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	✓	-	
mPLUG 2	-	✓	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-	
mPLUG-owl	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
VILBERT	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B2T2	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LXMERT	-	✓	✓	✓	✓	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
UNITER	-	✓	✓	✓	✓	✓	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-
Unicoder-VL	-	✓	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Pixel-BERT	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UNIMO [410]	-	✓	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UNIMO 2 [411]	-	✓	✓	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
BLIP	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
BLIP 2	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-
CLIP^2	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
ViLT	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InterBERT	-	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-
ImageBERT	-	✓	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓
OSCAR	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-
Virtex	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-
FLIP	-	✓	-	-	✓	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
COCA [422]	-	✓	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
LSeg	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
VL-BERT	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
VideoBERT	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ALBEF [423]	-	✓	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-	-
SimVLM [424]	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FILIP	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-	-
VLMo [425]	-	✓	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-	-
SOHO [426]	-	✓	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-
MAP [427]	-	✓	-	-	-	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-	-