A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT

YIHAN CAO[*], Lehigh University & Carnegie Mellon University, USA
SIYU LI, Lehigh University, USA
YIXIN LIU, Lehigh University, USA
ZHILING YAN, Lehigh University, USA
YUTONG DAI, Lehigh University, USA
PHILIP S. YU, University of Illinois at Chicago, USA
LICHAO SUN, Lehigh University, USA

Recently, ChatGPT, along with DALL-E-2 [1] and Codex [2],has been gaining significant attention from society. As a result, many individuals have become interested in related resources and are seeking to uncover the background and secrets behind its impressive performance. In fact, ChatGPT and other Generative AI (GAI) techniques belong to the category of Artificial Intelligence Generated Content (AIGC), which involves the creation of digital content, such as images, music, and natural language, through AI models. The goal of AIGC is to make the content creation process more efficient and accessible, allowing for the production of high-quality content at a faster pace. AIGC is achieved by extracting and understanding intent information from instructions provided by human, and generating the content according to its knowledge and the intent information. In recent years, large-scale models have become increasingly important in AIGC as they provide better intent extraction and thus, improved generation results. With the growth of data and the size of the models, the distribution that the model can learn becomes more comprehensive and closer to reality, leading to more realistic and high-quality content generation. This survey provides a comprehensive review on the history of generative models, and basic components, recent advances in AIGC from unimodal interaction and multimodal interaction. From the perspective of unimodality, we introduce the generation tasks and relative models of text and image. From the perspective of multimodality, we introduce the cross-application between the modalities mentioned above. Finally, we discuss the existing open problems and future challenges in AIGC.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun. 2018. A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT. *J. ACM* 37, 4, Article 111 (August 2018), 44 pages. https://doi.org/XXXXXXX.XXXXXXX

[*]Incoming Ph.D. student at Lehigh University.

Authors' addresses: Yihan Cao, yihanc@andrew.cmu.edu, Lehigh University & Carnegie Mellon University, Pittsburgh, PA, USA; Siyu Li, applicantlisiyu@hotmail.com, Lehigh University, Bethlehem, PA, USA; Yixin Liu, lis221@lehigh.edu, Lehigh University, Bethlehem, PA, USA; Zhiling Yan, zhilingyan724@outlook.com, Lehigh University, Bethlehem, PA, USA; Yutong Dai, lis221@lehigh.edu, Lehigh University, Bethlehem, PA, USA; Philip S. Yu, University of Illinois at Chicago, Chicago, Illinois, USA, psyu@uic.edu; Lichao Sun, lis221@lehigh.edu, Lehigh University, Bethlehem, PA, USA.

1 INTRODUCTION

In recent years, Artificial Intelligence Generated Content (AIGC) has gained much attention beyond the computer science community, where the whole society begins to be interested in the various content generation products built by large tech companies [3], such as ChatGPT [4] and DALL-E-2 [5]. AIGC refers to content that is generated using advanced Generative AI (GAI) techniques, as opposed to being created by human authors, which can automate the creation of large amounts of content in a short amount of time. For example, ChatGPT is a language model developed by OpenAI for building conversational AI systems, which can efficiently understand and respond to human language inputs in a meaningful way. In addition, DALL-E-2 is another state-of-the-art GAI model also developed by OpenAI, which is capable of creating unique and high-quality images from textual descriptions in a few minutes, such as "an astronaut riding a horse ina photorealistic style" as shown in Figure 1. As the remarkable achievements in AIGC, many people believe it will be the new era of AI and make significant impacts on the whole world.

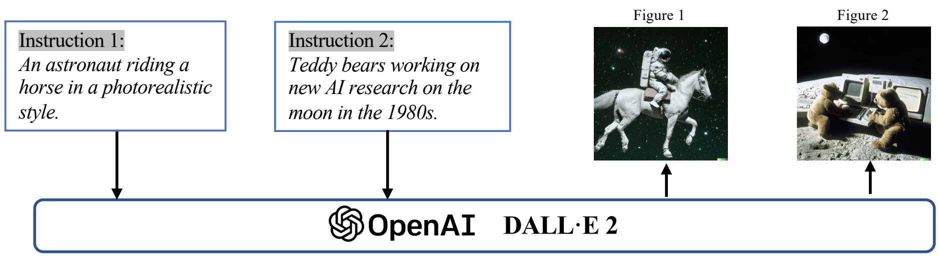
Instruction 1:
An astronaut riding a horse in a photorealistic style.
Instruction 2:
Teddy bears working on new AI research on the moon in the 1980s.
Figure 1
Figure 2
OpenAI DALL·E 2

Fig. 1. Examples of AIGC in image generation. Text instructions are given to OpenAI DALL-E-2 model, and it generates two images according to the instructions.

Technically, AIGC refers to, given human instructions which could help teach and guide the model to complete the task, utilizing GAI algorithms to generate content that satisfies the instruction. This generation process usually consists of two steps: extracting intent information from human instructions and generating content according to the extracted intentions. However, the paradigm of GAI models containing the above two steps is not entirely novel, as demonstrated by previous studies [6, 7]. The core advancements in recent AIGC compared to prior works are the result of training more sophisticated generative models on larger datasets, using larger foundation model architectures, and having access to extensive computational resources. For example, the main framework of GPT-3 maintains the same as GPT-2, but the pre-training data size grows from WebText [8](38GB) to CommonCrawl [9](570GB after filtering), and the foundation model size grows from 1.5B to 175B. Therefore, GPT-3 has better generalization ability than GPT-2 on various tasks, such as human intent extraction.

In addition to the benefits brought by the increase in data volume and computational power, researchers are also exploring ways to integrate new technologies with GAI algorithms. For example, ChatGPT utilizes reinforcement learning from human feedback (RLHF) [10–12] to determine the most appropriate response for a given instruction, thus improving model's reliability and accuracy over time. This approach allows ChatGPT to better understand human preferences in long dialogues. Meanwhile, in computer vision, stable diffusion [13], proposed by Stability.AI in 2022, has also shown great success in image generation. Unlike prior methods, generative diffusion models can help generate high-resolution images by controlling the trade-off between exploration and exploitation, resulting in a harmonious combination of diversity in the generated images and similarity to the training data.

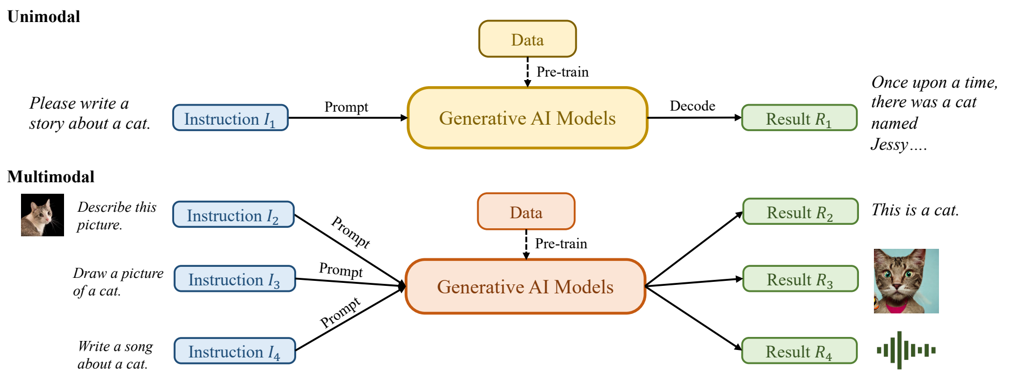

Fig. 2. Overview of AIGC. Generally, GAI models can be categorized into two types: unimodal models and multimodal models. Unimodal models receive instructions from the same modality as the generated content modality, whereas multimodal models accept cross-modal instructions and produce results of different modalities.

By combining these advancements, models have made significant progress in AIGC tasks and have been adopted in various industries, including art [14], advertising [15], and education [16]. In the near future, AIGC will continue to be a significant area of research in machine learning. It is therefore crucial to conduct an extensive review of past research and identify the open problems in this field. This survey is the first one that focuses on the core technologies and applications in the field of AIGC.

1.1 Major Contributions

This is the first comprehensive survey of AIGC that summarizes GAI in the aspects of techniques and applications. Previous surveys have focused on GAI from various angles, including natural language generation [17], image generation[18], generation in multimodal machine learning [7, 19]. However, these prior works only focus on a specific part of AIGC. In this survey, we first provide a review of foundation techniques commonly used in AIGC. Then, we further offer a thorough summary of advanced GAI algorithms, both in terms of unimodal generation and multimodal generation, as shown in Figure 2. In addition, we examine the applications and potential challenges of AIGC. Finally, we highlight the open problems and future directions in this field. In summary, the main contributions of this paper are as follows:

- To our best knowledge, we are the first to provide a formal definition and a thorough survey for AIGC and AI-enhanced generation process.
- We review the history, foundation techniques of AIGC and conduct a comprehensive analysis of recent advances in GAI tasks and models from the perspective of unimodal generation and multimodal generation.
- We discuss the main challenges facing AIGC and future research trends confronting AIGC.

1.2 Organization

The rest of the survey is organized as follows. Section 2 reviews the history of AIGC mainly from the view of vision and language modalities. Section 3 introduces the basic components that are widely used in nowadays GAI model training. Section 4 summarizes recent advances of GAI models, among which, Section 4.1 reviews the advances from unimodal perspective and Section 4.2 reviews the advances from the perspective of multimodal generation. Among multimodal generation, we introduce vision language models, text audio models, text graph models and text code models. Section 5 and Section 6 introduce the applications of GAI models in AIGC and some other important research that are related to this area. Furthermore, Sections 7, 8 reveal the risk, open problems and future directions of AIGC technologies. Finally, we conclude our research in 9.

2 HISTORY OF GENERATIVE AI

Generative models have a long history in artificial intelligence, dating back to the 1950s with the development of Hidden Markov Models (HMMs) [20] and Gaussian Mixture Models (GMMs) [21]. These models generated sequential data such as speech and time series. However, it wasn't until the advent of deep learning that generative models saw significant improvements in performance.

In early years of deep generative models, different areas do not have much overlap in general. In natural language processing (NLP), a traditional method to generate sentences is to learn word distribution using N-gram language modeling [22] and then search for the best sequence. However, this method cannot effectively adapt to long sentences. To solve this problem, recurrent neural networks (RNNs) [23] were later introduced for language modeling tasks , allowing for modeling relatively long dependency. This was followed by the development of Long Short-Term Memory (LSTM) [24] and Gated Recurrent Unit (GRU) [25], which leveraged gating mechanism to control memory during training. These methods are capable of attending to around 200 tokens in a sample [26], which marks a significant improvement compared to N-gram language models.

Meanwhile, in computer vision (CV), before the advent of deep learning-based methods, traditional image generation algorithms used techniques such as texture synthesis [27] and texture mapping [28]. These algorithms were based on hand-designed features, and were limited in their ability to generate complex and diverse images. In 2014, Generative Adversarial Networks (GANs) [29] was first proposed, which was a significant milestone in this area, due to its impressive results in various applications. Variational Autoencoders (VAEs) [30] and other methods like diffusion generative models [31] have also been developed for more fine-grained control over the image generation process and the ability to generate high-quality images.

The advancement of generative models in various domains has followed different paths, but eventually, the intersection emerged: the transformer architecture [32]. Introduced by Vaswani et al. for NLP tasks in 2017, Transformer has later been applied in CV and then become the dominant backbone for many generative models in various domains [9, 33, 34]. In the field of NLP, many prominent large language models, e.g., BERT and GPT, adopt the transformer architecture as their primary building block, offering advantages over previous building blocks, i.e., LSTM and GRU. In CV, Vision Transformer (ViT) [35] and Swin Transformer [36] later takes this concept even further by combining the transformer architecture with visual components, allowing it to be applied to image based downstreams. Except for the improvement that transformer brought to individual modalities, this intersection also enabled models from different domains to be fused together for multimodal tasks. One such example of multimodal models is CLIP [37]. CLIP is a joint vision-language model that combines the transformer architecture with visual components, allowing it to be trained on a massive amount of text and image data. Since it combines visual and language knowledge during pre-training, it can also be used as image encoders in multimodal prompting for generation. In all, the emergence of transformer based models revolutionized AI generation and led to the possibility of large-scale training.

In recent years, researchers have also begun to introduce new techniques based on these models. For instance, in NLP, instead of fine-tuning, people sometimes prefer few-shot prompting [38], which refers to including a few examples selected from the dataset in the prompt, to help the model better understand task requirements. And in visual language, researchers often combine

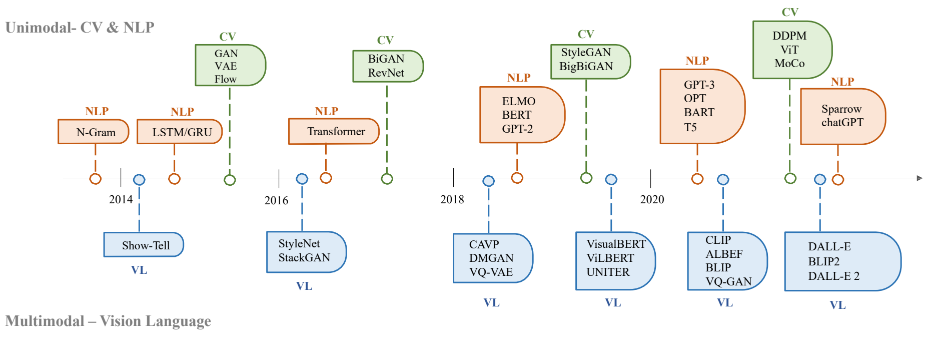

Fig. 3. The history of Generative AI in CV, NLP and VL.

modality-specific models with self-supervised contrastive learning objectives to provide more robust representations.

In the future, as AIGC becomes increasingly important, more and more technologies shall be introduced, empowering this area with vitality.

3 FOUNDATIONS FOR AIGC

In this section, we introduce foundation models that are commonly used in AIGC.

3.1 Foundation Model

3.1.1 Transformer. Transformer is the backbone architecture for many state-of-the-art models, such as GPT-3 [9], DALL-E-2 [5], Codex [2], and Gopher [39]. It was first proposed to solve the limitations of traditional models such as RNNs in handling variable-length sequences and context-awareness. Transformer architecture is mainly based on a self-attention mechanism that allows the model to attend to different parts in a input sequence. Transformer consists of an encoder and a decoder. The encoder takes in the input sequence and generates hidden representations, while the decoder takes in the hidden representation and generates output sequence. Each layer of the encoder and decoder consists of a multi-head attention and a feed-forward neural network. The multi-head attention is the core component of Transformer, which learns to assign different weights to tokens according their relevance. This information routing method allows the model to be better at handling long term dependency, hence, improving the performance in a wide range of NLP tasks. Another advantage of transformer is that its architecture makes it highly parallelizable, and allows data to trump inductive biases [40]. This property makes transformer well-suited for large-scale pre-training, enabling transformer based models to become adaptable to different downstream tasks.

3.1.2 Pre-trained Language Models. Since the introduction of the Transformer architecture, it has become the dominant choice in natural language processing due to its parallelism and learning capabilities. Generally, these transformer based pre-trained language models can be commonly classified into two types based on their training tasks: autoregressive language modeling and masked language modeling [41]. Given a sentence, which is composed of several tokens, the objective of masked language modeling, e.g., BERT [42] and RoBERTa [43], refers to predicting the probability of a masked token given context information. The most notable example of masked language modeling is BERT [42], which includes masked language modeling and next sentence

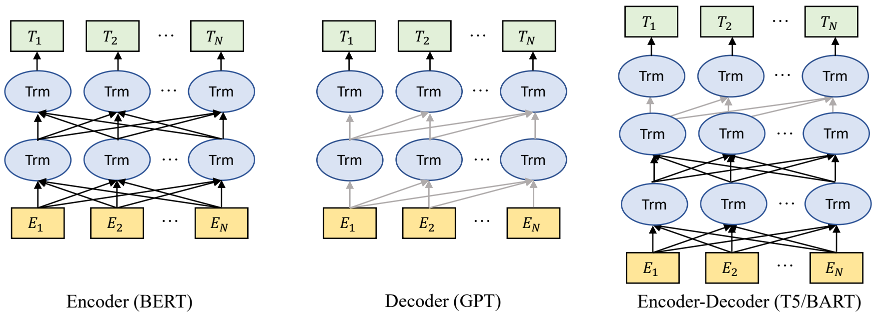

Fig. 4. Categories of pre-trained LLMs. Black line represents information flow in bidirectional models, while gray line representas left-to-right information flow. Encoder models, e.g. BERT, are trained with context-aware objectives. Decoder models, e.g. GPT, are trained with autoregressive objectives. Encoder-decoder models, e.g. T5 and BART, combines the two, which use context-aware structures as encoders and left-to-right structures as decoders.

prediction tasks. RoBERTa [43], which uses the same architecture as BERT, improves its performance by increasing the amount of pre-training data and incorporating more challenging pre-training objectives. XL-Net [44], which is also based on BERT, incorporates permutation operations to change the prediction order for each training iteration, allowing the model to learn more information across tokens. While autoregressive language modeling, e.g., GPT-3 [9] and OPT [45], is to model the probability of the next token given previous tokens, hence, left-to-right language modeling. Different from masked language models, autoregressive models are more suitable for generative tasks. We will introduce more about autoregressive models in Section 4.1.1.

3.2 Reinforcement Learning from Human Feedback

Despite being trained on large-scale data, the AIGC may not always produce output that aligns with the user's intent, which includes considerations of usefulness and truthfulness. In order to better align AIGC output with human preferences, reinforcement learning from human feedback (RLHF) has been applied to fine-tune models in various applications such as Sparrow, InstructGPT, and ChatGPT [10, 46].

Typically, the whole pipeline of RLHF includes the following three steps: *pre-training, reward learning, and fine-tuning with reinforcement learning.* First, a language model θ_0 is pre-trained on large-scale datasets as an initial language model. Since the *(prompt-answer)* pair given by θ_0 might not align with human purposes, in the second step we train a reward model to encode the diversified and complex human preference. Specifically, given the same prompt x, different generated answers $\{y_1, y_2, \cdots, y_3\}$ are evaluated by humans in a pairwise manner. The pairwise comparison relationships are later transferred to pointwise reward scalars, $\{r_1, r_2, \cdots, r_3\}$, using an algorithm such as ELO [47]. In the final step, the language model θ is fine-tuned to maximize the learned reward function using reinforcement learning. To stabilize the RL training, Proximal Policy Optimization (PPO) is often used as the RL algorithm. In each episode of RL training, an empirically-estimated KL penalty term is considered to prevent the model from outputting something peculiar to trick the reward model. Specifically, the total reward r_{total} at each step is given by $r_{total}(x, y) = r_{RM}(x, y) - \lambda_{\mathrm{KL}} D_{\mathrm{KL}} \left(\pi_\theta | \pi_{\theta_0}\right)$, where r_{RM} is the learned reward model, D_{KL} is the KL penalty term, and $\pi.$ is the trained policy. For more details on RLHF, please refer to [48].

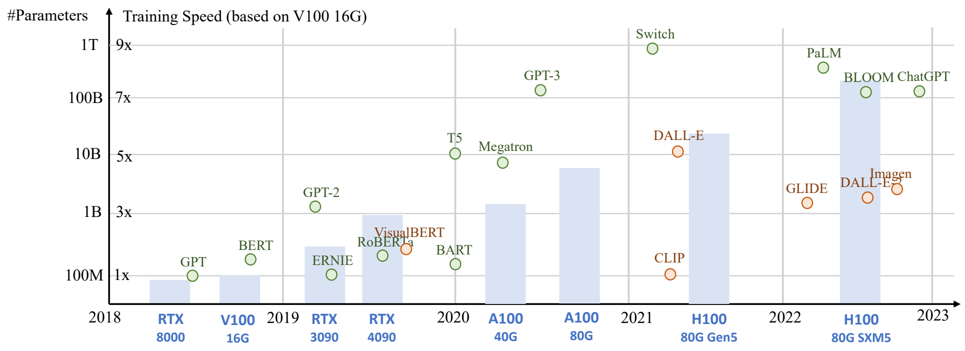

Fig. 5. Statistics of model size [52] and training speed [1]across different models and computing devices.

Although RLHF has shown promising results by incorporating fluency, progress in this field is impeded by a lack of publicly available benchmarks and implementation resources, leading to a perception that RL is a challenging approach for NLP. To address this issue, an open-source library named RL4LMs [49] has recently been introduced, consisting of building blocks for fine-tuning and evaluating RL algorithms on LM-based generation.

Beyond human feedback, the latest dialogue agent, Claude, favors Constitutional AI [50], where the reward model is learned via RL from AI Feedback (RLAIF). Both the critiques and the AI feedback are guided by a small set of principles drawn from a "constitution", which is the only thing provided by humans in Claude. The AI feedback focuses on controlling the outputs to be less harmful by explaining its objections to dangerous queries. Moreover, recently a preliminary theoretical analysis of the RLAIF [51] justifies the empirical success of RLHF and provides new insights for specialized RLHF algorithm design for language models.

3.3 Computing

3.3.1 Hardware. In recent years, there have been significant hardware advancements that have facilitated the training of large-scale models. In the past, training a large neural network using CPUs could take several days or even weeks. However, with the emergence of more powerful computing resources, this process has been accelerated by several orders of magnitude. For instance, the NVIDIA A100 GPU achieves seven times faster during BERT-large inference compared to the V100 and 11 times faster than the T4[2]. Additionally, Google's Tensor Processing Units (TPUs), which are designed specifically for deep learning, offer even higher computing performance compared to the current generation of A100 GPUs[3]. This rapid progress in computing power has significantly increased the efficiency of AI model training and opened up new possibilities for developing large and complex models.

3.3.2 Distributed training. Another significant improvement is distributed training. In traditional machine learning, training is typically performed on a single machine using a single processor. This approach can work well for small datasets and models, but it becomes impractical when

[1]https://lambdalabs.com/gpu-benchmarks
[2]https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf
[3]https://cloud.google.com/blog/products/ai-machine-learning/google-wins-mlperf-benchmarks-with-tpu-v4

dealing with large datasets and complex models. In distributed training, the training workload is split among multiple processors or machines, allowing the model to be trained much faster. Some companies have also released frameworks that simplify the process of distributed training on deep learning stacks [53–55]. These frameworks provide tools and APIs that allow developers to easily distribute their training workloads across multiple processors or machines, without having to manage the underlying infrastructure.

3.3.3 Cloud computing. Cloud computing has also played a vital role in training large-scale models. Previously, models are often trained locally. Now with the cloud computing services like AWS and Azure providing access to powerful computing resources, deep learning researchers and practitioners could spin up large clusters of GPUs or TPUs as needed for training large-scale models. Overall, these advancements have enabled the development of more complex and accurate models, unlocking new possibilities in various areas of AI research and applications.

4 GENERATIVE AI

4.1 Unimodal Models

In this section, we will introduce state-of-the-art unimodal generative models. These models are designed to accept a specific raw data modality as input, such as text or images, and then generate predictions in the same modality as the input. We will discuss some of the most promising approaches and techniques used in these models, including generative language models, e.g., GPT-3 [9], BART [34], T5 [56], and generative vision models, e.g., GAN [29], VAE [30], and normalizng flow [57].

4.1.1 Generative Language Models. Generative language models (GLMs) are a type of NLP models that are trained to generate readable human language based on patterns and structures in input data that they have been exposed to. These models can be used for a wide range of NLP tasks such as dialogue systems [58], , translation [59] and question answering [60].

Recently, The use of pre-trained language models has emerged as the prevailing technique in the domain of NLP. Generally, current state-of-the-art pre-trained language models could be categorized as masked language models (encoders), autoregressive language models (decoders) and encoder-decoder language models, as shown in Figure 4. Decoder models are widely used for text generation, while encoder models are mainly applied to classification tasks. By combining the strengths of both structures, encoder-decoder models can leverage both context information and autoregressive properties to improve performance across a variety of tasks. The primary focus of this survey is on generative models. In the following sections, we will delve into recent advancements in decoder and encoder-decoder architectures.

Decoder Models. One of the most prominent examples of autoregressive decoder-based language models is GPT [61], which is a transformer-based model that utilizes self-attention mechanisms to process all words in a sequence simultaneously. GPT is trained on next word prediction task based on previous words, allowing it to generate coherent text. Subsequently, GPT-2 [62] and GPT-3 [9] maintains the autoregressive left-to-right training method, while scaling up model parameters and leveraging diverse datasets beyond basic web text, achieving state-of-the-art results on numerous datasets. Gopher [39] uses a GPT-like structure but replace LayerNorm [63] with RSNorm, where a residual connection is added to the original layernorm structure to maintain the information. In addition to enhancing the normalization function, several other studies have concentrated on optimizing the attention mechanism. BLOOM [64] shares the same structure as GPT-3 but instead of using sparse attention, BLOOM uses a full attention network, which is better suited for modeling long dependencies. [65] proposes Megatron, which extends commonly used architectures like

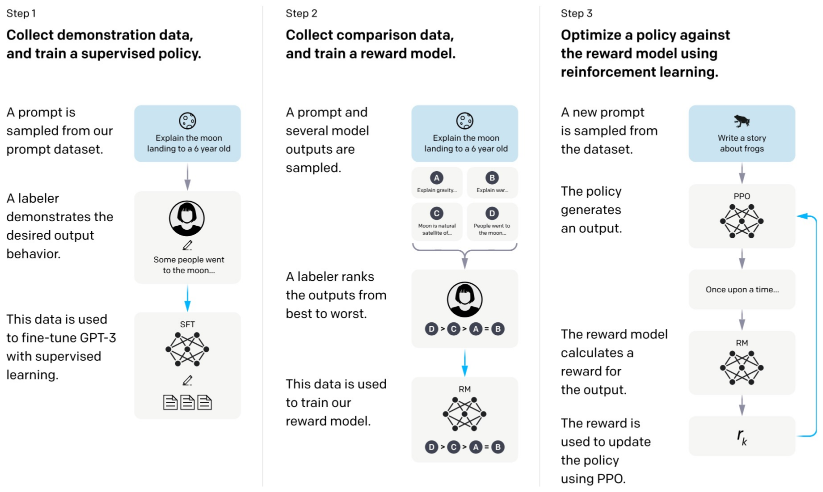
Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
Explain the moon landing to a 6 year old
A Explain gravity...
B Explain war...
C Moon is natural satellite of...
D People went to the moon...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
RM
D > C > A = B
Step 3
Optimize a policy against the reward model using reinforcement learning.
A new prompt is sampled from the dataset.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Fig. 6. The architecture of InstructGPT [10]. First, demonstration data are collected with human labelers and is used to fine-tune GPT-3. Then prompts and corresponding answers are sampled from the language model and human labelers will rank the answers from best to worst. This data is used to train a reward model. Finally, with the trained reward model, the language model could be optimized according to the preference of human labelers.

GPT-3, BERT and T5 with distributed training objectives to process large amount of data. This method is also later adopted by MT-NLG [66] and OPT [45]. Except for the advancements in model architecture and pre-training tasks, there has also been significant efforts put into improving the fine-tuning process for language models. For example, InstructGPT [10] takes advantage of pre-trained GPT-3 and uses RLHF for fine-tuning, allowing the model to learn preference according to ranking feedback labeled by human.

Encoder-Decoder Models. One of the main encoder-decoder methods is Text-to-Text Transfer Transformer (T5) [56], which combines transformer-based encoders and decoders together for pre-training. T5 employs a "text-to-text" approach, which means that it transforms both the input and output data into a standardized text format. This allows T5 to be trained on a wide range of NLP tasks, such as machine translation, question-answering, summarization, and more, using the same model architecture. Switch Transformer [67], as stated in its name, utilizes "switching", which refers to a simplified MoE routing algorithm, for parallelized training on T5. This model successfully obtained larger scale and better performance with the same computational resources compared to the base model. Another widely-used method that improves upon T5 is ExT5 [68], which is proposed by Google in 2021, extending the scale of previous T5 model. Compared to T5, ExT5 is continue pre-trained on C4 and ExMix, which is a combinition of 107 supervised NLP tasks across diverse domains. Another widely used encoder-decoder method is BART [34], which blends the bidirectional encoder from BERT and the autoregressive decoder from GPT, allowing it to leverage the bidirectional modeling abilities of the encoder while retaining the autoregressive properties for generation tasks. HTLM [69] leverages BART denoising objectives for modeling hyper-text language, which contains valuable information regarding document-level structure. This model also achieves state-of-the-art performance on zero-shot learning on various generation tasks.

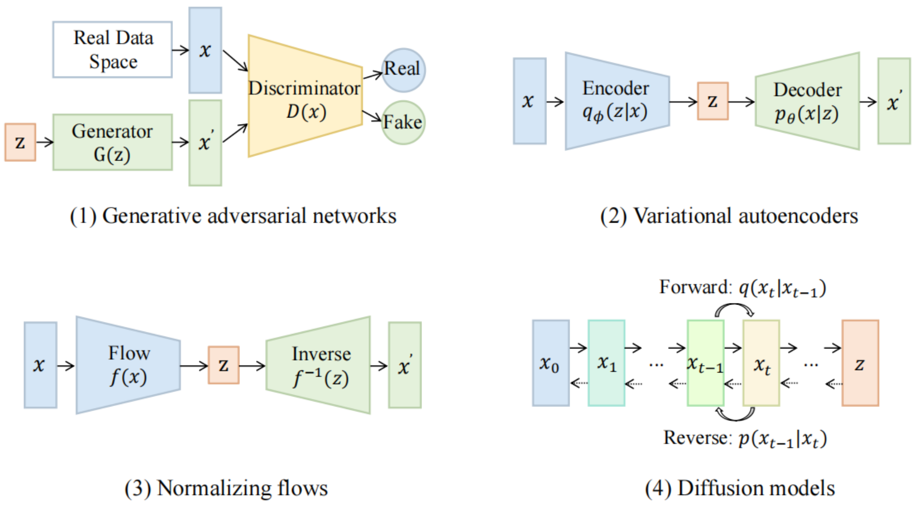

Fig. 7. Categories of vision generative models.

While DQ-BART [70], instead, aims at compressing BART into a smaller model using distillation and quantization, which achieves the BART original performance on various downstream tasks.

4.1.2 *Vision Generative Models.*

GAN. Generative Adversarial Networks (GANs) have gained popularity in the field of image generation research. GANs consist of two parts, a generator and a discriminator. The generator attempts to learn the distribution of real examples in order to generate new data, while the discriminator determines whether the input is from the real data space or not.

Structure. The structure of the generator and the discriminator highly influence GAN's training stability and performance. LAPGAN [71] generates high-quality images in a coarse-to-fine fashion using a cascade of convolutional networks within a Laplacian pyramid framework [72]. A. Radford et al. [73] propose DCGANs structure, a class of CNNs with architectural constraints, as a powerful solution for unsupervised learning. Progressive GAN [74] progressively grows the generator and discriminator, starting from low resolution and adding layers to model finer details, resulting in faster and more stable training and producing high-quality images. As traditional convolutional GANs generate high-resolution details based only on spatially local points in lower-resolution feature maps, SAGAN [75] introduces attention-driven, long-range dependency modeling and spectral normalization for improved training dynamics. In addition, generating high-resolution and diverse samples from complex datasets remains a challenge. To address this, BigGAN [76] is proposed as a large scale TPU implementation of GANs. StyleGAN [77] improves GANs by separating high-level attributes and variations, allowing for intuitive control and better performance in terms of quality metrics, interpolation, and disentanglement. [78, 79] focus on inverse mapping - projecting data back into the latent space, resulting in a useful feature representation for auxiliary discrimination tasks. To address mode collapse and improve the generative model, both the D2GAN [80] and GMAN [81] methods extend the traditional GANs by combining extra discriminators. MGAN [82] and MAD-GAN [83] address the mode collapse problem by incorporating multiple generators and one discriminator. CoGAN [84] is composed of a pair of GANs with a weight-sharing constraint, allowing for learning the joint distribution from separate marginal distributions without requiring corresponding images in the training set.

Representative variants. As the latent vector z of the generator is highly unstructured, Info-GAN [85] proposes another latent code c to extract the significant structured features of the actual data space. In CGANs [86–88], the generator and discriminator are conditioned on additional information, such as class labels or data from other modalities, to generate samples that are conditioned on specific attributes. f-GAN [89] allows for the use of any f-divergence as the objective function for training the generative model. The choice of f-divergence provides a flexible framework for controlling the trade-off between the quality of the generated samples and the difficulty of training the model.

Objective function. The goal of generative models is to match the real data distribution. WGAN [90] and LS-GAN [91, 92] aim to regularize the loss function with a Lipschitz regularity condition on the density of real data in order to better generalize and produce realistic new data. [93] is a weight normalization technique proposed to stabilize the training of the discriminator in GANs. Che et al. [94] regularize the objective, which can stabilize the training of GAN models. UGAN [95] stabilizes training of GANs by defining the generator objective with respect to an unrolled optimization of the discriminator. [96] makes discriminator relativistic by sampling from real/generated data pairs to improve stability and coverage of the data distribution generated by the generator.

VAE. Following variational bayes inference [97], Variational Autoencoders (VAE) are generative models that attempt to reflect data to a probabilistic distribution and learn reconstruction that is close to its original input.

Complex priors. Rewriting the variational evidence lower bound objective (ELBO) of variational autoencoders contributes to improve the variational bounds [98]. Since the true aggregate posterior is intractable, VampPrior [99] introduces a variational mixture of posteriors priors conditioned on learnable pseudo-inputs. [100–102] propose skip connections around the stochastic sampling process to capture different aspects of the data distribution.

Regularized Autoencoders. [1, 103, 104] introduce regularisation to the latent space of the encoder and lead to a smooth and representative latent space without conforming to an arbitrarily chosen prior. [105] propose a multi-scale hierarchical organization to model larger images.

Flow. A Normalizing Flow is a distribution transformation from simple to complex by a sequence of invertible and differentiable mappings.

Coupling and autoregressive flows. A non-linear deterministic transformation of the data is learned through a coupling method in [57] to make the transformed data conform to a factorized distribution. Dinh et al. [106] proposes multi-scale flow to gradually introduce dimensions to the distribution in the generative direction. A more flexible generalisation of coupling layers is the autoregressive flow [107–109], which permits parallel density estimation as a universal approximator.

Convolutional and Residual Flows. Zheng et al. [110] used 1D convolutions (ConvFlow) and Hoogeboom et al. [111] have provided a more general solution for modelling d×d convolutions. They exploited the triangular structure to improve the interaction among inputs and efficiently compute the determinant. RevNets [112] and iRevNets [113] are the first to build a reversible network architecture based on residual connections, which alleviate the vanishing gradients problem. In addition, the residual connections can be viewed as discretizations of a first order ordinary differential equation (ODE) [114] to improve parameter efficiency.

Diffusion. The Generative Diffusion Model (GDM) is a cutting-edge class of generative models based on probability, which demonstrates state-of-the-art results in the field of computer vision. It works by progressively corrupting data with multiple-level noise perturbations and then learning to reverse this process for sample generation.

Model Formulations. Diffusion Models are mainly formulated into three categories. DDPM [115] applies two Markov chains respectively to progressively corrupt data with Gaussian noise and reverse the forward diffusion process by learning Markov transition kernels. Score-based generative models (SGMs) directly work on the gradient of log density of data a.k.a score function. NCSN [31] perturbs data with multi-scale intensifying noise and jointly estimates score function of all such noisy data distribution by a neural network conditioned on all noise levels. It enjoys flexible sampling due to the completely decoupled training and inference steps. Score SDE [116] generalizes previous two formulations into continuous settings, where noise perturbations and denoising processes are solutions to stochastic differential equations. It is proved that probability flow ODE could also be used to model the reverse process.

Training Enhancement. Training enhancement aims to improve sampling by introducing prior knowledge from another pre-trained model or extra trainable hyper-parameters. Inspired from the idea of knowledge distillation, Salimans et al. [117] propose to progressively distill knowledge from a pre-trained complicated teacher model to a faster student model, which could cut sampling steps in half. TDPM [118] and ES-DDPM [119] improve sampling speed by truncating the diffusion process with early stop. To generate sample from reverse process initialized by a non-Gaussian distribution, another pre-trained generative model such as VAE or GAN is introduced to approximate such distribution. Franzese et al. [120] formulate the number of training steps as a variable to realize an optimal trade-off. Improved DDPM [121] first introduces noise scale tuning by adding noise scale term into loss function.Meanwhile, San Romans et al [122] introduce a noise prediction network to enable noise schedule adjustment step-by-step. Such noise schedule learning improves reconstruction by efficiently guiding the random walk of noise during training and inference.

Efficient Training-free Sampling. Instead of additional training, training-free sampling directly reduce the number of discretized time steps while minimizing discretization errors. Under same training objective, DDIM [123] generalizes DDPM to a class of non-Markovian diffusion process and introduces jump-step acceleration. This could provide shorter generative Markov chains. Analytic-DPM [124] provides more efficient inference by estimating the analytic form of optimal model reverse variance and KL-divergence w.r.t its score function. There are also works [125, 126] which directly figure out optimal sampling trajectories via dynamic programming.

Noise Distribution. The distribution of noise perturbations is an essential part of diffusion models and most of them are Gaussian. Meanwhile, fitting such distribution with more degrees of freedom could benefit performance. Nachmani et al. [127] prove that Gamma distribution could improve image and speech generation and a mixture of Gaussian distribution also outperforms a single distribution.Furthermore, cold diffusion [128] proposes a more generalized conclusion that noise can be set to any distribution as the generative behavior of diffusion model is not strongly dependent on the choice of noise distribution. Apart from noise perturbation, CCDF [129] shows it is unnecessary to initialize from Gaussian distribution and it could reduce sampling steps with a simple forward diffusion but better noise initialization.

Mixed Modeling. Mixed-modeling is aimed at combining diffusion model with another category of generative model to take all their advantages, which could provide stronger expressiveness or higher sampling speed. DiffuseVAE [130] merges a standard VAE into the DDPM pipeline by conditioning diffusion sampling process with blurry image reconstructions generated by VAE. LSGM [131] trains SGMs in the latent space of VAE, which generalizes SGMs into non-continuous data and enables smoother SGMs learning in a small space. Denoising diffusion GANs [132] introduces conditional GANs into DDPM pipeline to parameterize denoising process with a more expressive multimodal distribution, which provides large denoising steps. DiffFlow [133] integrates flow function into

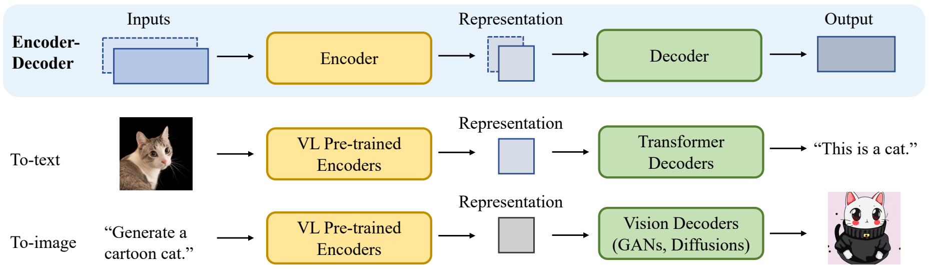

Fig. 8. The general structure of generative vision language. We separate the generation process into encoder part and decoder part. Encoder models will encode the inputs into a latent representation and then the decoder will decode this representation into a generated output.

trajectories of SDE-based diffusion model, which makes forward steps also trainable. The introduced randomness from noise perturbation endows normalizing flow with stronger expression power while the trainable forward process substantially reduce the diffusion trajectory length. Therefore, DiffFlow is able to learn distribution with sharper boundaries with better sampling efficiency.

4.2 Multimodal Models

Multimodal generation serves as an essential part in nowadays AIGC. The goal of multimodal generation is to learn a model that generates raw modalities by learning the multimodal connection and interaction from data [7]. This connection and interaction between modalities can sometimes be very intricate, which makes the multimodal representation space hard to learn compared to the unimodal one. However, with the emergence of the powerful modality-specific foundation architectures mentioned in previous sections, a growing number of methods are proposed in response to this challenge. In this section, we introduce the state-of-the-art multimodal models in vision language generation, text audio generation, text graph generation and text code generation. Since most multimodal generative models are always highly related to real-world applications, this section will mainly introduce from the perspective of downstream tasks.

4.2.1 Vision Language Generation. The encoder-decoder architecture is a widely used framework for solving unimodal generation problems in computer vision and natural language processing. In multimodal generation, particularly in vision-language generation, this method is often used as a foundation architecture. The encoder is responsible for learning a contextualized representation of the input data, while the decoder is used to generate raw modalities that reflect cross-modal interactions, structure, and coherence in the representation. In the following, we present a comprehensive survey of state-of-the-art vision-language encoders, followed by an exposition of the decoder component.

Vision Language Encoders. Recently, the development of encoders for single modalities has advanced significantly, leading to the question of how to learn contextualized representations from multiple modalities. A common way to do this is to combine modality-specific encoders using a fusion function and then leverage multiple pre-training tasks to align the representation space [37, 134, 135]. Generally. these encoder models could be separated into two categories, concatenated encoders and cross-aligned encoders [7].

Concatenated Encoders. A straight-forward solution to this problem is by concatenating the embeddings from single encoders. An early example is VisualBERT [134], which leverages BERT

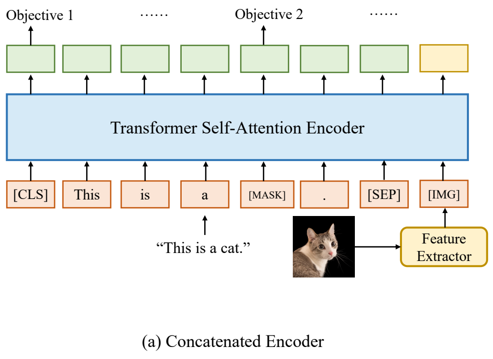

Fig. 9. Two types of vision language encoders: concatenated encoders and cross-aligned encoders. Concatenated encoders accepts concatenated embeddings from different raw modalities, and cross-aligned encoders are aligned in abstract modalities.

as text encoder, CNN as image encoder. The embeddings from the image encoder will be directly incorporated into BERT input embeddings, allowing the model to implicitly learn the aligned joint representation space. VisualBERT also leverages the multi-task pre-training paradigm as BERT, using two visually-grounded language model objectives: masked language modeling with image and sentence image prediction. Additionally, VisualBERT also incorporated some modality-specific pre-trianing objectives. Another example is VL-BERT [136], which shares the similar architecture as VisualBERT. Different from VisualBERT, VL-BERT uses Faster R-CNN [137] as regions of interest (ROI) extractor, and leverages this extracted ROI information as the image region embedding. VL-BERT also includes an additional pre-training task, masked ROI classification with linguistic clues, for better incoporating the visual information. Later, UNITER [138] was proposed based on the same architecture as VisualBERT, but with different training objectives. UNITER uses masked language modeling, masked region modeling, image text matching prediction and word region alignment prediction as its pre-training tasks. In this way, UNITER could learn informative contextualized embeddings. To this end, we see that concatenated encoders are generally based on the same BERT architecture, and pre-trained with BERT-like tasks. However, these models always involves a very complicated pre-training process, data collection and loss design. To solve this problem, [135] proposed SimVLM, which simplified the pre-training procedure of vision language models by setting PrefixLM as the training objective and directly using ViT as both text encoder and image encoder. SimVLM achieved state-of-the-art performance on multiple vision language tasks compared with previous methods with a much simplified architecture.

Cross-aligned Encoders. In addition to concatenating embeddings as input to encoders, another way to learn contextualized representations is to look at pairwise interactions between modalities [7]. Different from concatenated encoders, cross-aligned encoders always use a two-tower structure, where one tower for each modality and then learn a joint representation space using a cross-modality encoder. LXMERT [139] uses Transformers to extract image features and text features, and then adds a multimodal cross-attention module for coordination learning. The resulting output embeddings would be visual embeddings, language embeddings and multimodal embeddings. The model is also pre-trained with several multimodal tasks. Similarly, ViLBERT [140] leverages a cross-transformer module to align the two modalities. Given vision and language embeddings, the

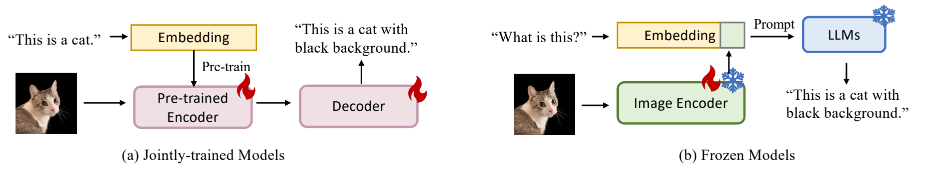

Fig. 10. Two types of to-language decoder models: jointly-trained models and frozen models. Jointly-trained models are normally trained end-to-end, while frozen models normally keep the language decoder frozen and only train the image encoder.

keys and values of one certain modality will be input into another modality's attention module to generate a pooled attention embedding that incorporates both information. In general, these models all leverage a cross layer to fuse the information into a joint representation space. Nevertheless, employing a transformer architecture in this context would be inefficient due to its large number of parameters. To simplify the training process and calculation, CLIP [37] uses dot product as the cross layer, which is more efficient than the transformer encoder, enabling efficient large-scale downstream training. Furthermore, CLIP is trained on copious amounts of pairwise data, which has been shown to outperform numerous other models.

Vision Language Decoders. Given a representation from a certain modality, vision language decoder mainly aims to transform it into a certain raw modality as specified by the task. In this section, we will mainly focus on to-text and to-image decoders.

To-text decoders. To-text decoders generally take in contextualized representations from the encoder and decode the representation into a sentence. With the emergence and proven effectiveness of large language models, many architectures are now selectively freezing the language decoder component. As a result, to-text decoders can be broadly categorized into two types: jointly-trained models and frozen models.

Jointly-trained decoders. Jointly-trained decoders refer to decoders that require complete cross-modal training when decoding representation. The challenge of text-to-text generation typically lies in aligning the two modalities during pre-training. As a result, the model requires a stronger encoder rather than a decoder. To address this challenge, many models prioritize constructing a strong encoder and then combine it with a relatively lightweight decoder model. For example, VLP [138] and ALBEF [141] leverage a simple transformer decoder to decode the information. BLIP [142] combines an encoder and decoder together during pre-training, allowing for multimodal space alignment for both understanding and generation objectives. BLIP is composed of three parts, a unimodal encoder for extracting image and text features, an image-grounded text encoder which accepts image and text features as input, and an image-grounded text decoder, which accepts image features and outputs text. Except for the aligned encoder and decoder structure, the authors also designed several corresponding pre-training tasks to help the model better learn the multimodal dependency.

Frozen deocders. Another way to efficiently perform to-text generation tasks is to freeze the large language model and train an image encoder only, which can also be seen as a way to perform multimodal prompting. Due to the success of prompting and in-context learning in NLP, there has been increased attention towards methods of this nature. This has led people to question whether such methods could be effective in multimodal settings as well. Frozen [143] first introduced in-context learning into vision language tasks. It freezes the language model and only trains the image encoder. The produced image representations will be embeded in the input embeddings of the language model. This method achieves state-of-the-art performance in various zero-shot and few-shot vision language tasks. Later, Alayrac et al. proposed Flamingo [144], which further explored multimodal in-context learning. Flamingo involves a frozen vision encoder and a frozen language encoder to get vision language representations, and utilizes gated cross-attention-dense layer to fuse the image representation into text representation. Recently, [145] proposed a method to realize VL dialogue with frozen language models, enabling the model to generate interleaved multimodal data. This method also freezes input encoders and train text-to-image and image-to-text linear maps to further encode and decode produced embeddings. However, it still remains a question why this kind of prompting based method work in multimodal generation. Some works have also been proposed to answer this question. Merullo et al. proposed a method [146] that injects a linear projection between the frozen image encoder and the text encoder. During training, only the linear projection is tuned. The experiment results show that frozen language models with similar sizes generally perform equally well at transferring visual information into language, but image encoders pre-trained with linguistic supervision like CLIP text encoder, could encode extra information and thus perform significantly better on vision language tasks.

To-image decoders. To-image generation refers to given an instruction, generating an image that corresponds to the instruction. Similarly, commonly used models in image generation also follow an encoder-decoder architecture, where the encoders are more focused on learning language information and the decoders are more focused on leveraging the learned information to restrict image synthesis. Generally, recent works could be separated into two categories, GAN-based methods and diffusion-based methods.

GAN-based decoders. Given a text encoder $\phi(t)$, GAN-based methods combine a discriminator D and a generator G, where the generator G accepts the text embedding generated by $\phi(t)$ and noise vector z to generate output X_g, which are input to the discriminator D with the real sample distribution X_r [147]. A notable model in this area is StackGAN [148]. StackGAN architecture consists of two stages: a conditioning stage and a refinement stage. In the conditioning stage, the model takes in the textual description as input and generates a low-resolution image. This image is then fed into the refinement stage, where it is further refined to produce a high-resolution image that matches the textual description. AttnGAN [149] is another text-to-image synthesis model that builds upon the StackGAN architecture. Attngan adds an attention mechanism to the StackGAN architecture to further improve the quality of generated images. However, these models mainly uses a comparatively simple text encoder during instruction learning, which could lead to certain information loss. StyleCLIP [150] is a recent model for text-to-image synthesis that uses contrastive learning to align text and image features. It is based on the StyleGAN [77] architecture and represents a significant advancement over previous text-to-image synthesis models such as StackGAN. StyleCLIP also follows the encoder-decoder structure that use a text encoder to encode instructions and an image decoder to synthesize a new image. One of the key innovations of StyleCLIP is its use of contrastive learning to align the text and image features. By training the model to maximize the similarity between the text and image features while minimizing the similarity between different text and image pairs, StyleCLIP is able to learn a more effective mapping between text and image features, resulting in higher-quality image synthesis.

Diffusion-based decoders. Generative image modelling has recently seen great success with the use of diffusion models. These models have also been applied in text-to-image generation. For example, GLIDE [151] introduces ablated diffusion model (ADM) into text-to-image generation. Compared to previous diffusion based methods, GLIDE uses larger model with 3.5B parameters and larger pairwise datasets, which achieved better results on many benchmarks. Different from GLIDE, Imagen [152] combines a frozen T5 language model with a super-resolution diffusion model.

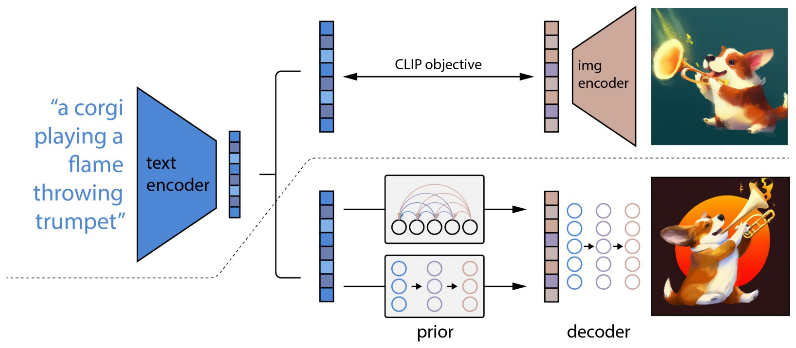

Fig. 11. The model structure of DALL-E-2. Above the dotted line is the CLIP pre-training process, which aims to align the vision and language modalities. And below the dotted line is the image generation process. The text encoder accepts an instruction and encodes it into a representation, then the prior network and diffusion model decodes this representation to generate the final output.

The frozen encoder will encode the text instruction and generates an embedding, then the first diffusion model will accordingly generate an low-resolution image. The second diffusion model accepts this image with the text embedding and outputs a high-resolution image. DALL-E-2 [5] combines CLIP encoder with diffusion decoder for image genration and editing tasks. Compared with Imagen, DALL-E-2 leverages a prior network to translation between text embedding and image embedding. Except for advancement in model design, another major difference between these diffusion based models and previous generative methods is that these diffusion based models are commonly trained on larger dataset with much more parameters, which make them possible to learn better representations over others.

In addition to previously mentioned methods, there are also works that use VAE as the decoder. For example, Ramesh et al. proposed DALL-E [33], a zero-shot image generator that utilizes dVAE as image encoder and decoder, BPE as text encoder and pre-trained CLIP during inference.

4.2.2 Text Audio Generation. The field of text-audio multimodal processing has seen significant growth in recent years. Most models in this field focus on either synthesis tasks, such as speech synthesis, or recognition tasks, such as automatic speech recognition. They refer to the process of converting written text into spoken speech or accurately transcribing human speech into machine-readable text. However, text audio generation is a distinct task that involves creating novel audio or text using multimodal models. While related, text-audio generation, synthesis, and recognition tasks differ in their goals and the techniques used to achieve them. In this work, we focus on text-audio generation rather than synthesis or recognition tasks.

Text-Audio Generation. AdaSpeech [153] is proposed to efficiently customize new voices with high quality using limited speech data by utilizing two acoustic encoders and conditional layer normalization in the mel-spectrogram decoder. Since previous studies have limitations in style conversion, Lombard [154] exploits the Spectral Shaping and Dynamic Range Compression [155] to generate highly intelligible speech in the presence of noise. Cross-lingual generation is another Influential work to transfer voices across languages. [156] can produce high-quality speech in multiple languages and transfer voices across languages through the use of phonemic input representation and adversarial loss term to disentangle speaker identity from speech content.

Text-Music Generation. [157] proposes a deep cross-modal correlation learning architecture for audio and lyrics, where intermodal canonical correlation analysis is used to calculate the similarity of temporal structures between audio and lyrics. To better learn social media content, JTAV [158] fuses textual, acoustic, and visual information using cross-modal fusion and attentive pooling techniques. Different from JTAV, [159] combines multiple types of information more related to music, such as playlists-track interactions and genre metadata, and align their latent representations to model unique music piece. In addition, there are some works focusing on generating text information, such as descriptions and captions, given the audio as input. [160] is proposed to generate descriptions for music playlists by combining audio content analysis and natural language processing to utilize the information of each track. MusCaps [161] is a music audio captioning model that generates descriptions of music audio content by processing audio-text inputs through a multimodal encoder and leveraging audio data pre-training to obtain effective musical feature representations. For music and language pre-training, Manco et al. [162] propose a multimodal architecture, which uses weakly aligned text as the only supervisory signal to learn general-purpose music audio representations. CLAP [163] is another method for learning audio concepts from natural language supervision that utilizes two encoders and contrastive learning to bring audio and text descriptions into a joint multimodal space.

4.2.3 Text Graph Generation. Text-graph generation is an essential multi-modal topic which could largely free the potential of NLP systems. Natural language text is intrinsically vague as it carries various redundant information and is also weakly organized in logic. Meanwhile, it is favorable for machines to work with structured, well-organized and compressed form of contents. Knowledge graph (KG) is structural meaning representation which reflects relationships among semantic internal states as graph structure in a language processing system. And there are increasing number of works extracting KG from text to assist text generation which incorporates complicated ideas across multiple sentences. Semantic parsing can also be formulated into a problem of text-to-graph generation. It aims to convert natural language text to a logical form, mostly abstract meaning representation (AMR) [164], which is a broad-coverage sentence-level semantic representation. Compared to text-to-KG generation, it emphasizes on providing machine interpretable representations rather than constructing a semantic network. Conversely, KG-to-text generation aims to generate fluent and logically-coherent text based on already constructed KG. Apart from the domain of NLP, text-graph generation could also push forward the boundary of computer aided drug design. There are emerging works bridging highly structured molecule graph with language descriptions, which facilitates human understanding of profound molecular knowledge and novel molecule exploration. In the following, we briefly overview some representative works in these four topics.

Text To Knowledge Graph Generation. Li et al. [165] treat text-to-KG construction as a process of knowledge graph completion (KGC), where missing terms are progressively covered by inference. A bilinear model and another DNN-based model are adopted to embed terms and compute score of arbitrary tuples for additive operation. KG-BERT [166] utilizes the power of pre-trained language models to capture more contextualized information during KGC. The idea is to represent triplets as textual sequences and models graph completion as a sequence classification problem by fine-tuned BERT model. Malaviya et al. [167] propose an approach incorporating graph convolutional network (GCN) for to extract more structural and semantic context. It also tackles graph sparsity and scalability issues by introducing graph augmentation and progressive masking strategies. Alternatively, another line of works [168–170] directly query pre-trained language models to obtain a semantic knowledge network. Specifically, language models are repeatedly prompted to predict the masked terms in cloze sentence to acquire relational knowledge. CycleGT [171] is an unsupervised method allowing text-KG translation in both directions. An unsupervised cycle training strategy is adopted to provide self-supervision, which enables the entire training process possible with non-parallel text and graph data. Utilizing similar strategy, DualTKB [172] further proves that model performance could be largely improved even under a weakly supervised setting. Lu et al. [173] propose a unified text-to-graph framework which incorporates most information extraction tasks. Meanwhile, the use of a pre-defined schema may limit its generalization to diverse text forms of nodes and edges. Grapher [174] performs end-to-end text-to-KG construction efficiently by generating node and edge in two separate stages. Specifically, a pre-trained language model is first fine-tuned with entity extraction tasks for node generation. Subsequently, focal loss and sparse adjacency matrix are introduced to address the skewed edge distribution issue during edge construction.

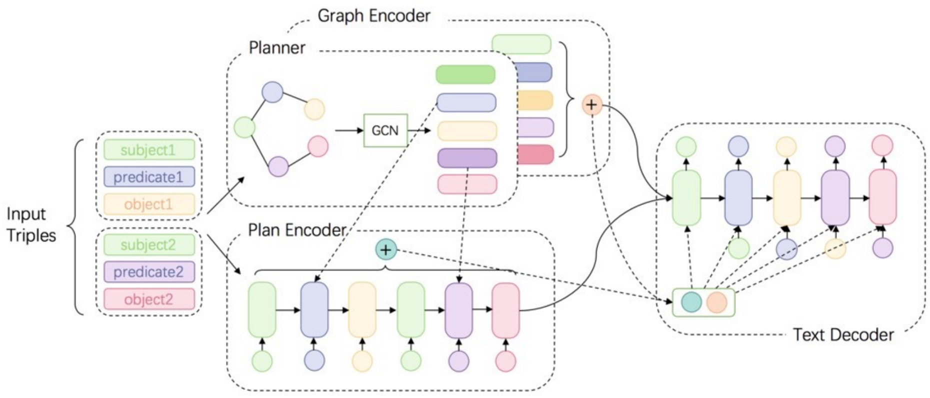

Fig. 12. DUALENC [175]: a KG-to-text generation model that bridges the structural gap between KG and graph via dual-encoding.

Knowledge Graph To Text Generation. GTR-LSTM [176] is a sequence-to-sequence encoder-decoder framework which generates text from linearized KG triples. It could handle cycles in KGs for capturing global information. Meanwhile, its linearized graph nature could still result in considerable structural information loss, especially for large graphs. To address this issue, Song et al. [177] encode graph semantics with a graph-state LSTM which enables information propagation between nodes during a series of state transitions. It proves to be capable of modeling non-local interactions between nodes while also efficient due to high parallelization. Zhao et al. [175] propose DUALENC, a dual encoding model, to bridge the structural discrepancy between input graph and output text. Specifically, it utilizes a GCN-based graph encoder to extract structural information, while a neural planner is also adopted to create a sequential content plan of a graph for generating linear output text. Alternatively, Koncel-Kedziorski et al. [178] encode graph structure for text generation with a transformer-based architecture extended from the graph attention network (GAT) [179]. The idea is to compute the node representations of KG by traversing its local neighborhood with self-attention mechanism. In contrast, Ribeiro et al. [180] focus on utilizing local and global node encoding strategies jointly to capture complementary information from graph contexts. Adapted from transformer, HetGT [181] aims at modeling different relationships in the graph independently to avoid information loss by simply mixing them. The input graph is first transformed into a heterogeneous Levi graph and then split into sub-graphs based on the heterogeneity of each part for future information aggregation.

Semantic Parsing. Early works [182, 183] formulate semantic parsing as sequence-to-sequence generation problems. However, AMR is a structured object by its nature. Sequence-to-sequence problem setup could only capture shallow word sequence information meanwhile potentially ignore abundant syntax and semantic information. Lyu et al. [184] model semantic parsing as a graph prediction problem by expressing AMR as a root labeled directed acyclic graph (DAG) This would require an alignment between node in the graph and word in the sentences. A neural parser which treat alignments as a latent variable in a joint probabilistic model is proposed for node alignment and edge prediction during AMR parsing. Chen et al. [185] construct semantic graph with an action set via a neural sequence-to-action RNN model. Parsing process are reinforced by integrating both structural and semantic constraints during decoding. Zhang et al. [186] tackle issues emerged from the reentrancy property in AMR parsing via an aligner-free attention based model which formulate the problem into sequence-to-graph transduction. Utilizing a pointer-generator network, it is proved that the model can be trained effectively with limited labeled AMR data. Fancellu et al. [187] propose a graph-aware sequential model to construct linearized graph for AMR graph prediction. Without a latent variable, it ensures each well-formed string will be only paired with exactly only one derivation by a novel graph-aware string rewriting strategy.

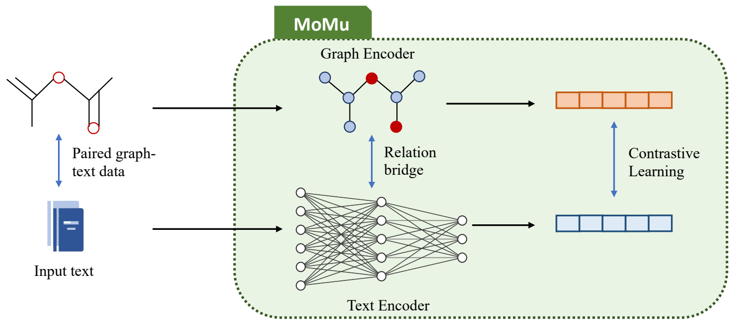

Fig. 13. MoMu [188]: A cross-modal text-molecule generation model.

Text Molecule Generation. Text2Mol [189] is a cross-modal information retrieval system to retrieve molecule graph based on language description. A BERT-based text encoder and a MLP-GCN combined molecule encoder are utilized to create multi-modal embedding in a semantic space, which is aligned by contrast learning with paired data. Instead of retrieving from existing molecules, MolT5 [190] proposes a self-supervised learning framework for text-conditioned de-novo molecule generation and molecule captioning. It tackles the scarcity of cross-modal data pair with a pre-train and fine-tune strategy. Specifically, it pre-trains the model on unpaired text and molecule strings with a denoising objective, followed by fine-tuning with limited paired data. However, restricted by its linearized graph nature, string-based representation of a molecule is not unique and could result in structural information loss. To tackle this issue, MoMu [188] introduces a graph based multi-modal framework which trains two separate encoders jointly by contrast learning for semantic space alignment with weakly-paired cross-modal data. It can also be adapted to various downstream tasks apart from de-novo molecule graph generation.

4.2.4 Text Code Generation.

Text-code generation aims to automatically generate valid programming code from natural language description or provide coding assist. LLMs have recently exhibited great potential in programming language (PL) code generation from natural language (NL) descriptions. Early works directly formulate text-code generation as a pure language generation task. However, NL and PL are data types with inherently different modalities, additional strategies are essential in capturing mutual dependencies between NL and PL during semantic space alignment. Compared to NL data, PL data also encapsulates rich structural information and different syntax, which makes it more challenging to understand semantic information from the PL context. Furthermore, text-code models are also expected to be multi-lingual as they could provide better generalization. In the following, we mainly introduce code generation models conditioned on NL description. We also review other coding assist models based on language.

Text-conditioned Programming Code Generation. CodeBERT [191] is a bimodal Transformer-based pre-trained text-code model which could capture the semantic connection between NL and PL. It adopts a hybrid objective function by utilizing binomial NL-PL paired data for model training and unimodal PL code data for learning better generators respectively to align NL and PL in semantic space. This model is further pre-trained on six multi-lingual PL for better generalization. CuBERT [192] shares similar model architecture with CodeBERT meanwhile it is not required to perform sentence separation between the natural-language description of a function and its body for sentence-pair representation. CodeT5 [193] proposes a pre-trained encoder-decoder Transformer model which better captures contextualized semantic information from code. Specifically, it introduces novel identifier-aware pre-training tasks to preserve crucial token type information by discriminating identifiers from code tokens and recover them when masked. PLBART [194] extends bimodal text-code model from generative tasks to a broader categories of discriminative tasks such as clone and vulnerable code detection under a unified framework. Another line of works [195, 196] introduce the notion of program graphs [197] to explicitly model the structures underlying PL code to assist generation. The program graphs are constructed as Abstract Syntax Trees (AST) to encapsulate knowledge from program-specific semantic and syntax.

Interactive Programming System. Text-code generation are jointly challenged by the intractable searching space of programming code generation and improper specification of user intent due to the intrinsic ambiguity of NL. CODEGEN [198] propose a multi-turn program synthesis approach which factorizes program synthesis conditioned on a single complicated NL specification into progressive generation controlled by a series of user intents. It is constructed in the form of autoregressive transformers learning a conditional distribution of the next token given previous tokens and it is trained on both PL and NL data. TDUIF [199] extends interactive programming framework by formalizing the user intent and providing more understandable user feedback. It further realizes scalable automatic algorithm evaluation which does not require user in loop with high-fidelity user interaction modeling.

5 APPLICATIONS

5.1 ChatBot

A chatbot is a computer program designed to simulate conversation with human users through text-based interfaces. Chatbots normally use language models to understand and respond to user queries and inpus in a conversational manner. They can be programmed to perform a wide range of tasks, for example, providing customer support and answering frequently asked questions. One

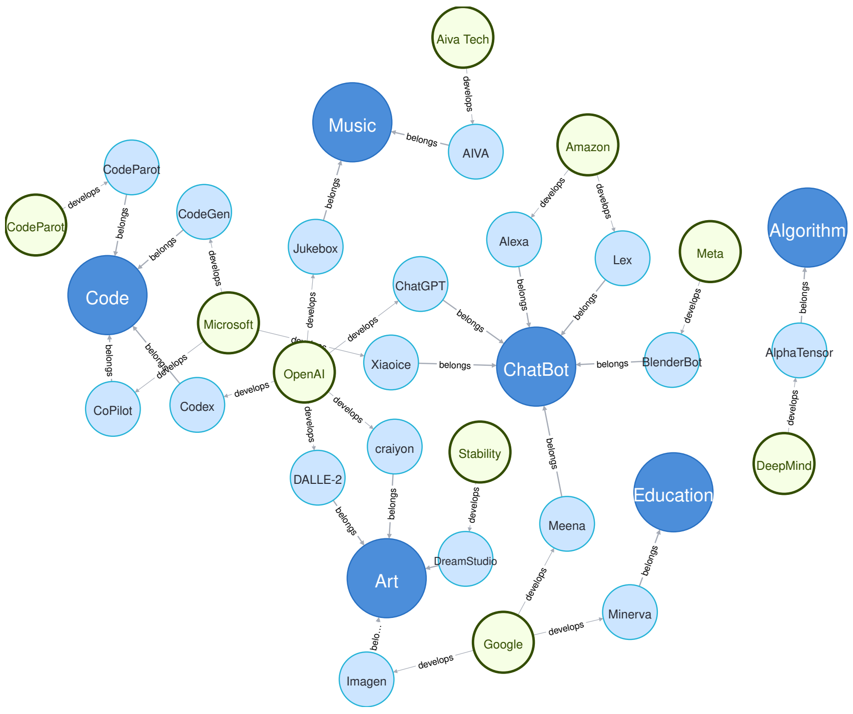

Fig. 14. A relation graph of a current research areas, applications and related companies, where dark blue circles represent research areas, light blue circles represent applications and green circles represents companies.

Application	**Platform/Software**	**Company**	**Year**	**Papaer**	**Link**
ChatBot	Xiaoice	Microsoft	2018	[200]	Xiaoice
ChatBot	Meena	Google	2020	[201]	Meena Blog
ChatBot	BlenderBot	Meta	2022	[202]	Blenderbot
ChatBot	ChatGPT	OpenAI	2022	[10]	ChatGPT
ChatBot	Alexa	Amazon	2014	-	Amazon Alexa
ChatBot	Lex	Amazon	2017	-	Amazon Lex
Music	AIVA	Aiva Tech	2016	-	AIVA
Music	Jukebox	OpenAI	2020	[203]	Jukebox
Code	CodeGPT	Microsoft	2021	[204]	CodeGPT
Code	CodeParrot	CodeParrot	2022	[205]	CodeParrot
Code	Codex	OpenAI	2021	[206]	Codex blog
Code	CoPilot	Microsoft	2021	[206]	CoPilot
Art	DALL-E-2	OpenAI	2022	[5]	DALL-E-2 Blog
Art	DreamStudio	Stability	2022	[13]	Dreamstudio
Art	craiyon	OpenAI	2021	[1]	Craiyon
Art	Imagen	Google	2022	[152]	Imagen
Education	Minerva	Google	2022	[207]	Minerva Blog
Algorithm	AlphaTensor	DeepMind	2022	[208]	AlphaTensor

Table 1. Applications of Generative AI models.

of the most prominent example is Xiaoice [200]. XiaoIce was developed by a team of researchers and engineers from Microsoft, using state-of-the-art techniques in natural language processing, machine learning, and knowledge representation. An important feature of Xiaoice is that it is able to express empathy, which is achieved by using sentiment analysis methods, to make Xiaoice perform like a human. In 2020, Google proposed Meena [201], a multi-turn open-domain chatbot trained on social media conversations, which achieves state-of-the-art interactive SSA score and perplexity. Recently, Microsoft released their newest version Bing, which incorporates ChatGPT, enabling its users to ask open domain or conditioned questions and get results through conversation. This presents new possibilities for the development of chatbots in the future.

5.2 Art

AI art generation refers to using computer algorithms to create original works of art. These algorithms are trained on large datasets of existing artwork and use machine learning techniques to generate new pieces that mimic the styles and techniques of famous artists or explore new artistic styles. With the rapid development in diffusion based models, more and more companies have launched their art generation products. One of the most notable advancements in the field is the DALL-E series, which was introduced by OpenAI. DALL-E [1], which is now Craiyon, was first built on VQ-VAE and CLIP, then diffusion was also applied to this product, becoming DALL-E-2 [5]. DreamStudio [13], created by Stability.ai, is a text-to-image generation service that utilizes stable diffusion to generate images based on given phrases or sentences. This technology offers comparable performance to that of DALL-E-2, but with even faster processing speeds, making it a popular choice for many users. Imagen [152], developed by Google, uses diffusion in its image editing and generation service. In a blog post, Google reported that they conducted a study with human raters to evaluate the quality of AI-generated images. The results showed that Imagen outperformed other models in side-by-side comparisons, with higher ratings for sample quality and image-text alignment preferred by the human raters.

5.3 Music

Deep music generation refers to the use of deep learning techniques and artificial intelligence algorithms to generate novel and original pieces of music. A prominent approach is to produce a symbolic representation of the music in the form of a piano roll. This approach entails specifying the timing, pitch, velocity, and instrument for each note to be played. AIVA [4] is one of the most notable examples, which is developed by Aiva Technologies in 2016. It can generate music clips in multiple styles including electronic, pop, jazz, etc. and can be used in various contexts. As the world's first artificial intelligence composer recognized by symphonic organizations, AIVA obtained the global status of Composer in the SACEM music society. OpenAI develops Jukebox [203] in 2020. It generates music with singing in the raw audio domain in diverse genres and artistic styles. Jukebox is considered as a leap forward in terms of musical quality, coherence, audio sample duration, and the capacity to be conditioned by artist, genre, and lyrics.

5.4 Code

AI-based programming systems generally aim for tasks including code completion, source code to pseudo-code mapping, program repair, API sequence prediction, user feedback, and natural language to code generation. Recently, the emergence of powerful LLMs has pushed the boundary of AI-based programming a large step forward. CodeGPT [204] is an open-source code generation model developed by OpenAI which follows the transformer architecture as many other models in the GPT

[4]http://www.aiva.ai

family. It can be fine-tuned for various code generation tasks such as code completion, summary, or translation based on a vast amount of source code data. CodeParrot [205] is a programming learning platform that provides user with personalized feedback and assistance during coding. A variety of interactive exercises and programming challenges are designed in the fashion of progressive human-machine interaction. One unique feature is the scaffolding strategy which splits complicated tasks into smaller and manageable steps to help students gradually build their coding skills. Trained on a much larger and more diverse corpus of data, Codex [206] is a significant step forward compared to most previous models. Specifically, it is designed to generate complete coding programs from scratch while CodeGPT is only able to generate code fragments that complete a given prompt. It also enjoys the benefits of being adapted to multiple programming languages, which could provide better flexibility and generalization.

5.5 Education

AIGC has the potential to achieve significant advancements in education by leveraging multi-modality data, for example, tutorial videos, academic papers, and other high-quality information, thereby improving the personalized education experience. On the academic side, Google Research introduced Minerva [207], which is built upon PaLM general language models [209] and an additional science-and-math-focused dataset, to solve college-level multi-step quantitative tasks, covering algebra, probability, physics, number theory, precalculus, geometry, biology, electric engineering, chemistry, astronomy, and machine learning. For example, it can give step-by-step details of proving the inequality $a^2 + b^2 \geq 2ab$ for any $(a, b) \in \mathbb{R}^2$ and it can also correctly identify Americium as the radioactive element among other three choices, including Sodium, Chromium, and Aluminum. As is described in the blog[5], Minerva achieves state-of-the-art performance on reasoning tasks by combing techniques, including few-shot prompting, a chain of thought or scratchpad prompting, and majority voting. Although Minerva's performance is still below human performance, with continuous improvement and future advancement, AIGC could provide affordable personalized math tutors. On the commercial side, Skillful Craftsman Education Technology announced to develop a class bot product powered by AIGC and featuring auto curriculum, AI tutor, and self-adaptive learning for online education, which is expected to be shipped by the fourth quarter of 2023.

6 EFFICIENCY IN AIGC

Deep generative AI models with neural networks has dominated the field of machine learning for the past decade, with its rise attributed to the ImageNet competition in 2012 [210], which led to a race to create deeper and more complex models. This trend is also seen in natural language understanding, where models like BERT and GPT-3 have been developed with a large number of parameters. However, the increasing model footprint and complexity, as well as the cost and resources required for training and deployment, pose challenges for practical deployment in the real world. The core challenge is efficiency, which can be broken it down as follows:

- **Inference efficiency**: This is concerned with the practical considerations of deploying a model for inference, i.e., computing the model's outputs for a given input. Inference efficiency is mostly related to the model's size, speed, and resource consumption (e.g., disk and RAM usage) during inference.
- **Training efficiency**: This covers factors that affect the speed and resource requirements of training a model, such as training time, memory footprint, and scalability across multiple

[5]https://ai.googleblog.com/2022/06/minerva-solving-quantitative-reasoning.html

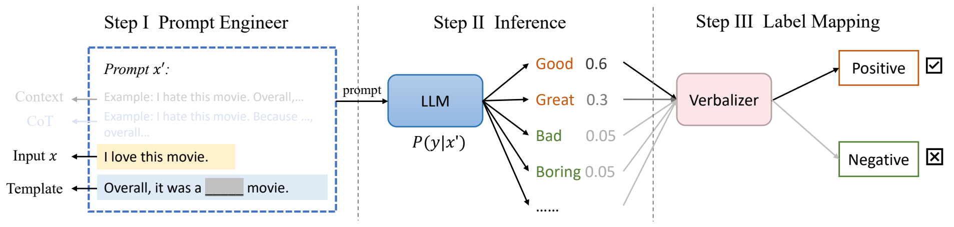

Fig. 15. General procedure of prompt learning for emotion detection examples. First, the user need to construct a prompt that fits the problem well, the user can also use in-context learning and chain-of-thought (CoT) to help improve the performance. Then, an LLM will generate suitable words for the blank space in the prompt. Finally, a verbalizer will project the generated word to a specific classification category.

devices. It may also encompass considerations around the amount of data required to achieve optimal performance on a given task.

6.1 Prompt Learning

Prompt learning is a relatively new concept that has been proposed in recent years within the context of pre-trained large language models. Previously, to make a prediction y given input x, the goal of traditional supervised learning is to find a language model that predicts the probability $P(y|x)$. With prompt learning, the goal becomes finding a template x' that directly predicts the probability $P(y|x')$ [211]. Hence, the objective of using a language model becomes encouraging a pre-trained model to make predictions by providing a prompt specifying the task to be done. Normally, prompt learning will freeze the language model and directly perform few-shot or zero-shot learning on it. This enables the language models to be pre-trained on large amount of raw text data and be adapted to new domains without tuning it again. Hence, prompt learning could help save much time and efforts.

6.1.1 Traditional Prompt Learning. The process of utilizing prompt learning with a language model can be divided into two main stages: prompt engineering and answer engineering.

- Prompt engineering. In general, there are two commonly used forms of prompt engineering: discrete prompt and continuous prompt. Discrete prompts are typically manually designed by humans for specific tasks, while continuous prompts are added to the input embeddings to convey task-specific information.
- Answer engineering. After the task has been reformulated, the answer generated by the language model based on the provided prompt needs to be mapped to the ground truth space. There are different paradigms for answering engineering, including discrete search space and continuous search space. As this topic is more closely related to classification tasks, we refer interested readers to for further information.

In addition to single-prompt learning methods, there are also multi-prompt methods. These approaches primarily focus on ensembling multiple prompts together as input during inference to improve prediction robustness, which is more effective than relying on a single prompt. Another approach to multi-prompt learning is prompt augmentation, which aims to assist the model in answering questions by providing additional prompts that have already been answered.

6.1.2 In-context Learning. Recently, in-context learning has received significant attention as an effective method for improving language models' performance. This approach is a subset of prompt learning and involves using a pre-trained language model as the backbone, along with adding a few input-label demonstration pairs and instructions to the prompt. In-context learning has been shown to be highly effective in guiding language models to produce better answers that are more closely aligned with the given prompt. Some recent studies have also suggested that in-context learning can be viewed as a form of implicit fine-tuning, as it enables the model to learn how to generate answers more accurately based on the input prompt.

6.2 Efficiency in Pretrained Foundation Models

Within the context of the AIGC framework, a fundamental component of each proposed method involves utilizing large pretrained foundation models (PFMs) [212]. PFMs, such as BERT [42], GPT-2 [62], and RoBERTa [43], have revolutionized the field of natural language processing by achieving state-of-the-art results on a wide range of NLP tasks. However, these models are incredibly large and computationally expensive, which can lead to efficiency problems. This is especially true when working with limited computational resources, such as on personal computers or in cloud environments with limited processing power. In order to address these efficiency problems, recent numerous works have been dedicated to exploring more cost-effective pretraining methods to pretrain large-scale PFMs. The effectiveness of learning algorithms is contingent upon both training methods and model architecture efficiency. For example, ELECTRA [213] introduces an RTD task that predicts whether each input marker is replaced by other tokens, thereby enabling ELECTRA to train against all input tokens. In addition to effective training methods, model architecture efficiency can also contribute to improved PFMs efficiency. Most PFMs based on the Transformer algorithm may benefit from a more efficient model architecture by reducing the complexity of the Transformer algorithm.

6.3 Model Compression

Model compression is an effective approach to reduce model size and improve computation efficiency. It requires fewer computing resources and memory and can better meet the needs of various applications than the original model, where its strategies can be divided into two categories: parameter compression and structure compression. Parameter compression methods include parameter pruning, parameter quantization, low-rank decomposition, and parameter sharing. Parameter pruning deletes redundant parameters based on a sizeable PFM, while parameter quantization reduces model parameters to lower-order numbers without significant impact on model performance. Low-rank decomposition reduces the dimension of a high-dimensional parameter vector, and parameter sharing maps model parameters to reduce their number. Structure compression refers to designing new compact network structures and employing knowledge distillation, where the knowledge learned from a larger teacher model is transferred to a smaller student model through soft labels, among other techniques. DistilBERT [214], for instance, uses knowledge distillation to compress BERT, reducing its size by 40% while maintaining 97% of its language comprehension. ALBERT uses decomposition-embedded parameterization and cross-layer parameter sharing to reduce the number of model parameters.

7 TRUSTWORTHY & RESPONSIBLE AIGC

While AIGC has the potential to be incredibly useful in many different applications, it also raises significant concerns about security and privacy. In this section, we will discuss studies that disclose the "dark" side of AIGC and countermeasures proposed to ensure that AIGC can be used in a safe and responsible way.

7.1 Security

Factuality. Although tools like ChatGPT [4] is capable of generating content that usually appears or sounds reasonable, they are often unreliable in terms of factuality [215]. Sometimes, the model outputs counterfactual or even absurd answers, which pose a serious threat to the truthfulness of information on the internet. Recently, NewsGuard's Misinformation Monitor [216] has indicated the possibility that AI-generated content tools are being weaponized to spread misinformation at an unprecedented scale. Presented with 100 samples from NewsGuard's proprietary misinformation database, the tested model, ChatGPT, generated false narratives for 80 of the 100 previously identified false arguments, which could easily come across as legitimate and authoritative for those unfamiliar with the topics [216]. Moreover, Alex [217] offers a more specific example by demonstrating how to leverage ChatGPT [4] to generate a newspaper. Besides natural language processing, factuality concerns also exist in the computer vision domain. For example, stable diffusion [13], which has been demonstrated to be a powerful vision-generated model, has trouble drawing realistic human hands with the correct number of fingers [218]. To prevent the spread of misinformation on the internet, websites like Stackoverflow [219] propose policies that ban users from using AI-generated content as an answer to reduce the risk of being overwhelmed by inaccurate and biased content.

Earlier studies have shown that AI models suffer from factual incorrectness and hallucination of knowledge [220]. To evaluate and improve the factual accuracy of AI-generated content, [221] proposed model-based metrics that measure the factualness of generated text, complementing traditional metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [222] and BLEU (Bilingual Evaluation Understudy) [223]. Specifically, [221] proposed a Transformer-based end-to-end fact extraction model, which enables the structured prediction of relation tuples for factualness assessment. More systematic definitions of truthfulness standards and approaches for governing AI-generated content were later proposed in Truthful AI [224]. The standard proposed by Truthful AI aims to avoid "negligent falsehoods" and explicitly train AI systems to be truthful via curated datasets and human interaction. Based on GPT-3, WebGPT [225] proposed a humanoid prototype that models the AI answering process into web searching and evidence-composing phrases. Since the model is trained to cite its sources, the factual accuracy of AI-generated content is significantly improved in multiple benchmark datasets [226, 227]. Specifically, the model is obtained by fine-tuning GPT-3 using imitation learning, which leverages human feedback to optimize answer quality. Furthermore, [228] measures and improves the factual accuracy of large-scale language models for open-ended text generation. [228] proposed the *factual-nucleus* sampling algorithm that dynamically adapts the randomness to balance the factuality and quality of AI-generated content. A *factuality-enhanced* training method that uses *TOPICPREFIX* for better awareness of facts and sentence completion is designed as the training objective, which vastly reduces factual errors. Despite these preliminary advances in developing more truthful AI, challenges still remain. For example, AI-generated content might be problematic on unfamiliar types of questions and contexts that involve contradictions [215].

Toxicity. Besides utility, it is important for AI-generated content (AIGC) to be helpful, harmless, unbiased, and non-toxic. Extensive research has been conducted on the potential harm caused by deployed models [229–231], which can include biased outputs [232, 233], stereotypes [234], and misinformation [235]. To address this issue of toxicity in the language domain, OpenAI proposes InstructGPT [10], which aligns language models with human preferences by using human feedback as a reward signal to fine-tune the models, ensuring more relevant and safe responses. Concurrently, Google proposes LaMDA [236], a family of neural language models specialized for safe and factual dialog by leveraging fine-tuning and external knowledge sources. To improve model safety, LaMDA [236] designs a set of metrics (Appendix A.1 in the original paper) that quantify model safety based on an illustrative set of human values derived from Google's AI Principles [6]. Furthermore, Ganguli *et al.* [237] study and improve the safety of language models in an adversarial way. Specifically, they investigate the scaling behaviors for red teaming across models with different sizes (2.7B, 13B, and 52B parameters) and training schemes (plain LM, fine-tuned LM, LM with rejection sampling, and LM trained with RLHF). They found that models trained with RLHF scale better and are increasingly difficult to red team.

7.2 Privacy

Membership inference. The goal of the membership inference attack (MIA) is to determine whether an image x belongs to the set of training data. Wu *et al.* [238] investigated the membership leakage in text-to-image (diffusion-based and sequence-to-sequence-based) generation models under realistic black-box settings. Specifically, three kinds of intuitions including quality, reconstruction error, and faithfulness are considered to design the attack algorithms. However, Wu *et al.* [238] assumed that the member set and the hold-out set come from different distributions, which makes the MIA much easier. Under a more practical setting [239], where the member set and the hold-out set are in the same distribution, Duan *et al.* [240] propose Step-wise Error Comparing Membership Inference (SecMI), a black-box MIA that infers memberships by assessing the matching of forward process posterior estimation at each timestep. Concurrently, Hu and Pang [241] propose two attack approaches, including loss-based and likelihood-based MIA. Furthermore, Matsumoto *et al.* [242] introduce more comparisons with GANs.

Data Extraction. The objective of a data extraction attack is to retrieve an image from the set of training data, denoted as $x \in D$. The attack can be considered a success if the attacker is able to obtain an image $\hat{x}$ that closely resembles image $x \in D$. Compared to the membership inference attack, the data extraction attack poses stronger privacy risks to the model. The feasibility of such an attack might be due to the memorization property of large-scale models [243], in which they turn to memorize parts of their training data. When prompted appropriately, the memorized training data that might contain sensitive information will be emitted verbatim. Earlier, in the language domain, Carlini *et al.* [244] demonstrated that large language models (specifically, GPT-2 [245]) memorize and leak individual training examples. Specifically, they proposed a simple and efficient method for extracting verbatim sequences from a language model's training set using only black-box query access. Recently, in the vision domain, Somepalli *et al.* [246] showed that the data replication problem existed in diffusion models, where the generated images are close to the training data in terms of semantic similarity. To disclose worse-case privacy risk, Carlini *et al.* [247] further explored the privacy vulnerabilities of state-of-the-art diffusion models by leveraging a generate-and-filter pipeline to extract over a thousand training examples from the models. Specifically, the extraction approach first samples 500 candidate images by querying the generation function in a black-box manner using selected prompts. Based on the intuition that generations of memorized data are nearly identical, a similarity graph is then constructed to determine whether an image belongs to the training set. The results in [247] show that diffusion models, including Stable Diffusion [13] and Imagen [152], are more susceptible to privacy breaches compared to earlier generative models like GANs [29]. These results highlight the necessity of developing new techniques for preserving privacy during training to address these vulnerabilities.

8 OPEN PROBLEMS AND FUTURE DIRECTIONS

In this section, we discuss some challenges in AIGC and potential ways to address them.

[6]https://ai.google/principles/

High-stakes Applications. Although the community has witnessed the huge success of AIGC in images, texts, and audio generations, these areas are arguably more fault-tolerant. On the contrary, AIGC for high-stakes applications, including healthcare [248], financial services [249], autonomous vehicles [250], and science discovery [251], are still challenging. In these domains, tasks are mission-critical and require a high degree of accuracy, reliability, transparency, and less or near zero fault-tolerant. For example, the large language model Galactica [252], which is made for automatically organizing science, can perform knowledge-intensive scientific tasks and have promising performances on several benchmark tasks. Its public demo were taken down from the service only three days after its initial release, due to the intensive criticism on its generated biased and incorrect results in an authoritative tone. It would be crucial for generative models in these high-stakes applications to give confidence scores, reasoning, and source information along with generated results. Only when professionals understand how and where these results are coming from, they can confidently utilize these tools in their tasks.

Specialization and Generalization. AIGC relies on the choice of foundation models, which are trained on different datasets, including crawl-based [37] one and carefully curated [252]. And it is argued in [230] that "training on a more diverse dataset is not always better for downstream performance than a more specialized foundation model." However, the curation of highly specialized dataset can be both time-consuming and cost-ineffective. A better understanding of cross-domain representations and how they are resilient to testing-time distribution-shift may guide the design of training datasets that balance specialization and generalization.

Continual Learning and Retraining. The human knowledge base keeps expanding and new tasks continue emerging. To generate the contents with up-to-date information, it not only requires model to "remember" the learned knowledge, but also be able to learn and infer from newly acquired information. For some scenarios [253], it suffices to perform the continual learning on downstream tasks while keeping the pre-trained foundation model unchanged. When necessary [254], one can perform continual learning on foundation models. However, it is also observed that the continual learning may not always outperform the retrained models [255]. This calls for the need to understand when should one choose continual learning strategy and when to choose to the retraining strategy. Also, training foundation models from scratch may be prohibitive, so modularized design of next generation of foundation models for AIGC may elucidate which parts of the model should be retrained.

Reasoning. Reasoning is a crucial component of human intelligence that enables us to draw inferences, make decisions, and solve complex problems. However, even trained with large scale dataset, sometimes GAI models could still fail at common sense reasoning tasks [256, 257]. Recently, more and more researchers began to focus on this problem. Chain-of-thought (CoT) prompting [256] is a promising solution to the challenge of reasoning in generative AI models. It is designed to enhance the ability of large language models to learn about logical reasoning in the context of question answering. By explaining the logical reasoning process that humans use to arrive at answers to models, they can follow the same road that humans take in processing their reasoning. By incorporating this approach, large language models can achieve higher accuracy and better performance in tasks that require logical reasoning. CoT has also been applied to other areas like vision language question answering [257] and code generation [258]. However, it still remains a problem that how to construct these CoT prompts according to specific tasks.

Scaling up. Scaling up has been a common problem in large-scale pretraining. Model training is always limited by compute budget, available dataset and model size. As the size of pretraining models increases, the time and resources required for training also increases significantly. This poses a challenge for researchers and organizations that seek to utilize large-scale pretraining for various tasks, such as natural language understanding, computer vision, and speech recognition.

Another issue pertains to the efficacy of pretraining with large-scale datasets, which may not yield optimal results if experimental hyperparameters, such as model size and data volume, are not thoughtfully designed. As such, suboptimal hyperparameters can result in wasteful resource consumption and the failure to achieve desired outcomes through further training. Several works have been proposed to solve these problems. Hoffmann et al. [259] introduce a formal scaling law to predict model performance based on the number of parameters and dataset size. This work provides a useful framework for understanding the relationship between these key factors when scaling up. Aghajanyan et al. [260] conduct empirical analyses to validate the Hoffmann scaling law and propose an additional formula that explores the relationship between different training tasks in multimodal model training settings. These findings provide valuable insights into the complexities of large-scale model training and the nuances of optimizing performance across diverse training domains.

Social issues. As AIGC continues to proliferate across various domains, social concerns regarding its use have become increasingly prominent. These concerns relate to issues such as bias, ethics, and the impact of AI-generated content on various stakeholders. One major concern is the potential for bias in AI-generated content, particularly in areas such as natural language processing and computer vision. AI models can inadvertently perpetuate or amplify existing societal biases, particularly if the training data used to develop the models are themselves biased. This can have significant negative consequences, such as perpetuating discrimination and inequities in areas such as hiring, loan approvals, and criminal justice. Ethical concerns also arise with the use of AI-generated content, particularly in cases where the technology is used to generate deepfakes or other forms of manipulated media. Such content can be used to spread false information, incite violence, or harm individuals or organizations. Additionally, there are concerns around the potential for AI-generated content to infringe on copyright and intellectual property rights, as well as issues around privacy and data security. Overall, while AI-generated content holds significant promise in various domains, it is crucial to address these social concerns to ensure that its use is responsible and beneficial for society as a whole.

9 CONCLUSION

This survey provides a comprehensive overview of the history and recent advancements in AIGC, with a particular focus on both unimodality and multimodality generative models. In addition, we discuss the recent applications of generative AI models, commonly used techniques in AIGC, and address concerns surrounding trustworthiness and responsibility in the field. Finally, we explore open problems and future directions for AIGC, highlighting potential avenues for innovation and progress. The primary objective of this survey is to provide readers with a comprehensive understanding of recent developments and future challenges in generative AI. Our analysis of the general framework of AI generation aims to distinguish contemporary generative AI models from their predecessors. Ultimately, we hope this survey will aid readers in gaining deeper insights into this field. Moving forward, we would further investigate this topic and provide a more comprehensive analysis of AIGC.

REFERENCES

[1] 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*, pp. 8821–8831, PMLR, 2021.

[2] 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.

[3] L. Yunjiu, W. Wei, and Y. Zheng, "Artificial intelligence-generated and human expert-designed vocabulary tests: A comparative study," *SAGE Open*, vol. 12, no. 1, p. 21582440221082130, 2022.

[4] "Chatgpt: Optimizing language models for dialogue," Nov. 2022.

[5] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," Apr. 2022. arXiv:2204.06125 [cs].

[6] M. Stefanini, M. Cornia, L. Baraldi, S. Cascianelli, G. Fiameni, and R. Cucchiara, "From Show to Tell: A Survey on Deep Learning-based Image Captioning," Nov. 2021. arXiv:2107.06912 [cs].

[7] P. P. Liang, A. Zadeh, and L.-P. Morency, "Foundations and Recent Trends in Multimodal Machine Learning: Principles, Challenges, and Open Questions," Sept. 2022. arXiv:2209.03430 [cs].

[8] A. Gokaslan and V. Cohen, "Openwebtext corpus," 2019.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[10] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," Mar. 2022. arXiv:2203.02155 [cs].

[11] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep Reinforcement Learning from Human Preferences," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[12] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano, "Learning to summarize from human feedback," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, (Red Hook, NY, USA), pp. 3008–3021, Curran Associates Inc., Dec. 2020.

[13] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," Apr. 2022. arXiv:2112.10752 [cs].

[14] N. Anantrasirichai and D. Bull, "Artificial intelligence in the creative industries: a review," *Artificial Intelligence Review*, vol. 55, pp. 589–656, jul 2021.

[15] J. Kietzmann, J. Paschen, and E. Treen, "Artificial Intelligence in Advertising," *Journal of Advertising Research*, vol. 58, no. 3, pp. 263–267, 2018. Publisher: Journal of Advertising Research _eprint: https://www.journalofadvertisingresearch.com/content/58/3/263.full.pdf.

[16] M. Kandlhofer, G. Steinbauer, S. Hirschmugl-Gaisch, and P. Huber, "Artificial intelligence and computer science in education: From kindergarten to university," in *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9, 2016.

[17] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, "Pretrained language models for text generation: A survey," 2022.

[18] J. Agnese, J. Herrera, H. Tao, and X. Zhu, "A survey and taxonomy of adversarial neural networks for text-to-image synthesis," 2019.

[19] M. Suzuki and Y. Matsuo, "A survey of multimodal deep generative models," *Advanced Robotics*, vol. 36, pp. 261–278, feb 2022.

[20] K. Knill and S. Young, "Hidden Markov Models in Speech and Language Processing," in *Corpus-Based Methods in Language and Speech Processing* (S. Young and G. Bloothooft, eds.), pp. 27–68, Dordrecht: Springer Netherlands, 1997.

[21] D. A. Reynolds *et al.*, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[22] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, p. 1137–1155, mar 2003.

[23] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model.," in *Interspeech*, vol. 2, pp. 1045–1048, Makuhari, 2010.

[24] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[25] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597–1600, IEEE, 2017.

[26] U. Khandelwal, H. He, P. Qi, and D. Jurafsky, "Sharp nearby, fuzzy far away: How neural language models use context," 2018.

[27] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1033–1038, IEEE, 1999.

[28] P. S. Heckbert, "Survey of texture mapping," *IEEE computer graphics and applications*, vol. 6, no. 11, pp. 56–67, 1986.

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[31] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach,

R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
[33] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," 2021.
[34] 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.
[35] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[36] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.
[37] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.
[38] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, L. Li, and Z. Sui, "A survey on in-context learning," 2023.
[39] 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.
[40] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, "A mathematical framework for transformer circuits," *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.
[41] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020.
[42] 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.
[43] 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.
[44] 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.
[45] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "Opt: Open pre-trained transformer language models," 2022.
[46] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, *et al.*, "Improving alignment of dialogue agents via targeted human judgements," *arXiv preprint arXiv:2209.14375*, 2022.
[47] R. Coulom, "Whole-history rating: A bayesian rating system for players of time-varying strength," in *Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29-October 1, 2008. Proceedings 6*, pp. 113–124, Springer, 2008.
[48] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.
[49] R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi, and Y. Choi, "Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization," *arXiv preprint arXiv:2210.01241*, 2022.
[50] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, *et al.*, "Constitutional ai: Harmlessness from ai feedback," *arXiv preprint arXiv:2212.08073*, 2022.
[51] B. Zhu, J. Jiao, and M. I. Jordan, "Principled reinforcement learning with human feedback from pairwise or k-wise comparisons," *arXiv preprint arXiv:2301.11270*, 2023.
[52] X. Amatriain, "Transformer models: an introduction and catalog," *arXiv preprint arXiv:2302.07730*, 2023.
[53] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," 2018.
[54] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3505–3506, 2020.
[55] J. J. Dai, D. Ding, D. Shi, S. Huang, J. Wang, X. Qiu, K. Huang, G. Song, Y. Wang, Q. Gong, *et al.*, "Bigdl 2.0: Seamless scaling of ai pipelines from laptops to distributed cluster," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21439–21446, 2022.
[56] 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.
[57] L. Dinh, D. Krueger, and Y. Bengio, “Nice: Non-linear independent components estimation,” *arXiv preprint arXiv:1410.8516*, 2014.
[58] J. Ni, T. Young, V. Pandelea, F. Xue, and E. Cambria, “Recent advances in deep learning based dialogue systems: A systematic survey,” *Artificial intelligence review*, pp. 1–101, 2022.
[59] S. Yang, Y. Wang, and X. Chu, “A survey of deep learning techniques for neural machine translation,” *arXiv preprint arXiv:2002.07526*, 2020.
[60] F. Zhu, W. Lei, C. Wang, J. Zheng, S. Poria, and T.-S. Chua, “Retrieving and reading: A comprehensive survey on open-domain question answering,” *arXiv preprint arXiv:2101.00774*, 2021.
[61] A. Radford and K. Narasimhan, “Improving language understanding by generative pre-training,” 2018.
[62] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” 2019.
[63] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.
[64] 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.
[65] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, “Megatron-lm: Training multi-billion parameter language models using model parallelism,” 2019.
[66] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro, “Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model,” 2022.
[67] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” *J. Mach. Learn. Res*, vol. 23, pp. 1–40, 2021.
[68] V. Aribandi, Y. Tay, T. Schuster, J. Rao, H. S. Zheng, S. V. Mehta, H. Zhuang, V. Q. Tran, D. Bahri, J. Ni, *et al.*, “Ext5: Towards extreme multi-task scaling for transfer learning,” *arXiv preprint arXiv:2111.10952*, 2021.
[69] A. Aghajanyan, D. Okhonko, M. Lewis, M. Joshi, H. Xu, G. Ghosh, and L. Zettlemoyer, “Htlm: Hyper-text pre-training and prompting of language models,” 2021.
[70] Z. Li, Z. Wang, M. Tan, R. Nallapati, P. Bhatia, A. Arnold, B. Xiang, and D. Roth, “Dq-bart: Efficient sequence-to-sequence model via joint distillation and quantization,” *arXiv preprint arXiv:2203.11239*, 2022.
[71] E. L. Denton, S. Chintala, R. Fergus, *et al.*, “Deep generative image models using a laplacian pyramid of adversarial networks,” *Advances in neural information processing systems*, vol. 28, 2015.
[72] P. J. Burt and E. H. Adelson, “The laplacian pyramid as a compact image code,” in *Readings in computer vision*, pp. 671–679, Elsevier, 1987.
[73] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
[74] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.
[75] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” in *International conference on machine learning*, pp. 7354–7363, PMLR, 2019.
[76] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.
[77] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.
[78] J. Donahue, P. Krähenbühl, and T. Darrell, “Adversarial feature learning,” *arXiv preprint arXiv:1605.09782*, 2016.
[79] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “It takes (only) two: Adversarial generator-encoder networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.
[80] T. Nguyen, T. Le, H. Vu, and D. Phung, “Dual discriminator generative adversarial nets,” *Advances in neural information processing systems*, vol. 30, 2017.
[81] I. Durugkar, I. Gemp, and S. Mahadevan, “Generative multi-adversarial networks,” *arXiv preprint arXiv:1611.01673*, 2016.
[82] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, “Multi-generator generative adversarial nets,” *arXiv preprint arXiv:1708.02556*, 2017.
[83] A. Ghosh, V. Kulharia, V. P. Namboodiri, P. H. Torr, and P. K. Dokania, “Multi-agent diverse generative adversarial networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8513–8521, 2018.
[84] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” *Advances in neural information processing systems*, vol. 29, 2016.
[85] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets,” *Advances in neural information processing systems*,

vol. 29, 2016.
[86] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[87] Y. Lu, Y.-W. Tai, and C.-K. Tang, "Attribute-guided face generation using conditional cyclegan," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 282–297, 2018.
[88] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1429–1437, 2019.
[89] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," *Advances in neural information processing systems*, vol. 29, 2016.
[90] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.
[91] G.-J. Qi, "Loss-sensitive generative adversarial networks on lipschitz densities," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1118–1140, 2020.
[92] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.
[93] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.
[94] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016.
[95] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.
[96] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018.
[97] C. W. Fox and S. J. Roberts, "A tutorial on variational bayesian inference," *Artificial intelligence review*, vol. 38, pp. 85–95, 2012.
[98] M. D. Hoffman and M. J. Johnson, "Elbo surgery: yet another way to carve up the variational evidence lower bound," in *Workshop in Advances in Approximate Bayesian Inference, NIPS*, vol. 1, 2016.
[99] J. Tomczak and M. Welling, "Vae with a vampprior," in *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223, PMLR, 2018.
[100] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther, "Biva: A very deep hierarchy of latent variables for generative modeling," *Advances in neural information processing systems*, vol. 32, 2019.
[101] A. Vahdat and J. Kautz, "Nvae: A deep hierarchical variational autoencoder," *Advances in neural information processing systems*, vol. 33, pp. 19667–19679, 2020.
[102] B. Wu, S. Nair, R. Martin-Martin, L. Fei-Fei, and C. Finn, "Greedy hierarchical variational autoencoders for large-scale video prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2318–2328, 2021.
[103] P. Ghosh, M. S. Sajjadi, A. Vergari, M. Black, and B. Schölkopf, "From variational to deterministic autoencoders," *arXiv preprint arXiv:1903.12436*, 2019.
[104] A. Van Den Oord, O. Vinyals, *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.
[105] A. Razavi, A. Van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Advances in neural information processing systems*, vol. 32, 2019.
[106] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.
[107] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," *Advances in neural information processing systems*, vol. 30, 2017.
[108] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, "Neural autoregressive flows," in *International Conference on Machine Learning*, pp. 2078–2087, PMLR, 2018.
[109] N. De Cao, W. Aziz, and I. Titov, "Block neural autoregressive flow," in *Uncertainty in artificial intelligence*, pp. 1263–1273, PMLR, 2020.
[110] G. Zheng, Y. Yang, and J. Carbonell, "Convolutional normalizing flows," *arXiv preprint arXiv:1711.02255*, 2017.
[111] E. Hoogeboom, R. Van Den Berg, and M. Welling, "Emerging convolutions for generative normalizing flows," in *International Conference on Machine Learning*, pp. 2771–2780, PMLR, 2019.
[112] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," *Advances in neural information processing systems*, vol. 30, 2017.
[113] J.-H. Jacobsen, A. Smeulders, and E. Oyallon, "i-revnet: Deep invertible networks," *arXiv preprint arXiv:1802.07088*, 2018.

[114] E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun, "Learning across scales—multiscale methods for convolution neural networks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[115] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[116] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[117] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[118] H. Zheng, P. He, W. Chen, and M. Zhou, "Truncated diffusion probabilistic models," *stat*, vol. 1050, p. 7, 2022.

[119] Z. Lyu, X. Xu, C. Yang, D. Lin, and B. Dai, "Accelerating diffusion models via early stop of the diffusion process," *arXiv preprint arXiv:2205.12524*, 2022.

[120] G. Franzese, S. Rossi, L. Yang, A. Finamore, D. Rossi, M. Filippone, and P. Michiardi, "How much is enough? a study on diffusion times in score-based generative models," 2022.

[121] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International Conference on Machine Learning*, pp. 8162–8171, PMLR, 2021.

[122] R. San-Roman, E. Nachmani, and L. Wolf, "Noise estimation for generative diffusion models," *arXiv preprint arXiv:2104.02600*, 2021.

[123] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[124] F. Bao, C. Li, J. Zhu, and B. Zhang, "Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models," *arXiv preprint arXiv:2201.06503*, 2022.

[125] D. Watson, J. Ho, M. Norouzi, and W. Chan, "Learning to efficiently sample from diffusion probabilistic models," *arXiv preprint arXiv:2106.03802*, 2021.

[126] D. Watson, W. Chan, J. Ho, and M. Norouzi, "Learning fast samplers for diffusion models by differentiating through sample quality," in *International Conference on Learning Representations*, 2022.

[127] E. Nachmani, R. S. Roman, and L. Wolf, "Non gaussian denoising diffusion models," 2021.

[128] A. Bansal, E. Borgnia, H.-M. Chu, J. S. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, "Cold diffusion: Inverting arbitrary image transforms without noise," *arXiv preprint arXiv:2208.09392*, 2022.

[129] H. Chung, B. Sim, and J. C. Ye, "Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction," 2021.

[130] K. Pandey, A. Mukherjee, P. Rai, and A. Kumar, "Diffusevae: Efficient, controllable and high-fidelity generation from low-dimensional latents," 2022.

[131] A. Vahdat, K. Kreis, and J. Kautz, "Score-based generative modeling in latent space," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11287–11302, 2021.

[132] Z. Xiao, K. Kreis, and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion gans," *arXiv preprint arXiv:2112.07804*, 2021.

[133] Q. Zhang and Y. Chen, "Diffusion normalizing flow," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16280–16291, 2021.

[134] 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.

[135] 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.

[136] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, "Vl-bert: Pre-training of generic visual-linguistic representations," *arXiv preprint arXiv:1908.08530*, 2019.

[137] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[138] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and vqa," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 13041–13049, 2020.

[139] H. Tan and M. Bansal, "Lxmert: Learning cross-modality encoder representations from transformers," *arXiv preprint arXiv:1908.07490*, 2019.

[140] J. Lu, D. Batra, D. Parikh, and S. Lee, *ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks.* Red Hook, NY, USA: Curran Associates Inc., 2019.

[141] 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.

[142] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International Conference on Machine Learning*, pp. 12888–12900, PMLR, 2022.

[143] M. Tsimpoukelli, J. Menick, S. Cabi, S. Eslami, O. Vinyals, and F. Hill, "Multimodal few-shot learning with frozen language models," *Proc. Neural Information Processing Systems*, 2021.

[144] 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," *arXiv preprint arXiv:2204.14198*, 2022.

[145] J. Y. Koh, R. Salakhutdinov, and D. Fried, "Grounding language models to images for multimodal generation," *arXiv preprint arXiv:2301.13823*, 2023.

[146] J. Merullo, L. Castricato, C. Eickhoff, and E. Pavlick, "Linearly mapping from image to text space," *arXiv preprint arXiv:2209.15162*, 2022.

[147] R. Zhou, C. Jiang, and Q. Xu, "A survey on generative adversarial network-based text-to-image synthesis," *Neurocomputing*, vol. 451, pp. 316–336, 2021.

[148] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017.

[149] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1316–1324, 2018.

[150] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski, "Styleclip: Text-driven manipulation of stylegan imagery," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2085–2094, 2021.

[151] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," 2021.

[152] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *arXiv preprint arXiv:2205.11487*, 2022.

[153] M. Chen, X. Tan, B. Li, Y. Liu, T. Qin, S. Zhao, and T.-Y. Liu, "Adaspeech: Adaptive text to speech for custom voice," *arXiv preprint arXiv:2103.00993*, 2021.

[154] D. Paul, M. P. Shifas, Y. Pantazis, and Y. Stylianou, "Enhancing speech intelligibility in text-to-speech synthesis using speaking style conversion," *arXiv preprint arXiv:2008.05809*, 2020.

[155] T.-C. Zorila, V. Kandia, and Y. Stylianou, "Speech-in-noise intelligibility improvement based on spectral shaping and dynamic range compression," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[156] Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Z. Chen, R. Skerry-Ryan, Y. Jia, A. Rosenberg, and B. Ramabhadran, "Learning to speak fluently in a foreign language: Multilingual speech synthesis and cross-language voice cloning," *arXiv preprint arXiv:1907.04448*, 2019.

[157] Y. Yu, S. Tang, F. Raposo, and L. Chen, "Deep cross-modal correlation learning for audio and lyrics in music retrieval," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 1, pp. 1–16, 2019.

[158] H. Liang, H. Wang, J. Wang, S. You, Z. Sun, J.-M. Wei, and Z. Yang, "Jtav: Jointly learning social media content representation by fusing textual, acoustic, and visual features," *arXiv preprint arXiv:1806.01483*, 2018.

[159] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched music representations with multiple cross-modal contrastive learning," *IEEE Signal Processing Letters*, vol. 28, pp. 733–737, 2021.

[160] K. Choi, G. Fazekas, B. McFee, K. Cho, and M. Sandler, "Towards music captioning: Generating music playlist descriptions," *arXiv preprint arXiv:1608.04868*, 2016.

[161] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Muscaps: Generating captions for music audio," in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2021.

[162] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Learning music audio representations via weak language supervision," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 456–460, IEEE, 2022.

[163] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "Clap: Learning audio concepts from natural language supervision," *arXiv preprint arXiv:2206.04769*, 2022.

[164] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, "Abstract meaning representation for sembanking," in *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pp. 178–186, 2013.

[165] X. Li, A. Taheri, L. Tu, and K. Gimpel, "Commonsense knowledge base completion," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1445–1455, 2016.

[166] L. Yao, C. Mao, and Y. Luo, "Kg-bert: Bert for knowledge graph completion," *arXiv preprint arXiv:1909.03193*, 2019.

[167] C. Malaviya, C. Bhagavatula, A. Bosselut, and Y. Choi, "Commonsense knowledge base completion with structural and semantic context," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 2925–2933, 2020.

[168] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?," *arXiv preprint arXiv:1909.01066*, 2019.

[169] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *arXiv preprint arXiv:2010.15980*, 2020.
[170] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.
[171] Q. Guo, Z. Jin, X. Qiu, W. Zhang, D. Wipf, and Z. Zhang, "Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training," 2020.
[172] P. L. Dognin, I. Melnyk, I. Padhi, C. N. dos Santos, and P. Das, "Dualtkb: A dual learning bridge between text and knowledge base," 2020.
[173] Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, and H. Wu, "Unified structure generation for universal information extraction," *arXiv preprint arXiv:2203.12277*, 2022.
[174] I. Melnyk, P. Dognin, and P. Das, "Knowledge graph generation from text," *arXiv preprint arXiv:2211.10511*, 2022.
[175] C. Zhao, M. Walker, and S. Chaturvedi, "Bridging the structural gap between encoding and decoding for data-to-text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 2481–2491, Association for Computational Linguistics, July 2020.
[176] B. Distiawan, J. Qi, R. Zhang, and W. Wang, "Gtr-lstm: A triple encoder for sentence generation from rdf data," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1627–1637, 2018.
[177] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "A graph-to-sequence model for amr-to-text generation," 2018.
[178] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text generation from knowledge graphs with graph transformers," *arXiv preprint arXiv:1904.02342*, 2019.
[179] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
[180] L. F. Ribeiro, Y. Zhang, C. Gardent, and I. Gurevych, "Modeling global and local node contexts for text generation from knowledge graphs," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 589–604, 2020.
[181] S. Yao, T. Wang, and X. Wan, "Heterogeneous graph transformer for graph-to-sequence learning," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7145–7154, 2020.
[182] L. Dong and M. Lapata, "Language to logical form with neural attention," *arXiv preprint arXiv:1601.01280*, 2016.
[183] R. Jia and P. Liang, "Data recombination for neural semantic parsing," *arXiv preprint arXiv:1606.03622*, 2016.
[184] C. Lyu and I. Titov, "Amr parsing as graph prediction with latent alignment," *arXiv preprint arXiv:1805.05286*, 2018.
[185] B. Chen, L. Sun, and X. Han, "Sequence-to-action: End-to-end semantic graph generation for semantic parsing," *arXiv preprint arXiv:1809.00773*, 2018.
[186] S. Zhang, X. Ma, K. Duh, and B. V. Durme, "Amr parsing as sequence-to-graph transduction," 2019.
[187] F. Fancellu, S. Gilroy, A. Lopez, and M. Lapata, "Semantic graph parsing with recurrent neural network dag grammars," *arXiv preprint arXiv:1910.00051*, 2019.
[188] B. Su, D. Du, Z. Yang, Y. Zhou, J. Li, A. Rao, H. Sun, Z. Lu, and J.-R. Wen, "A molecular multimodal foundation model associating molecule graphs with natural language," 2022.
[189] C. Edwards, C. Zhai, and H. Ji, "Text2mol: Cross-modal molecule retrieval with natural language queries," pp. 595–607, 01 2021.
[190] C. Edwards, T. Lai, K. Ros, G. Honke, K. Cho, and H. Ji, "Translation between molecules and natural language," 2022.
[191] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, *et al.*, "Codebert: A pre-trained model for programming and natural languages," *arXiv preprint arXiv:2002.08155*, 2020.
[192] A. Kanade, P. Maniatis, G. Balakrishnan, and K. Shi, "Learning and evaluating contextual embedding of source code," in *International conference on machine learning*, pp. 5110–5121, PMLR, 2020.
[193] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," *arXiv preprint arXiv:2109.00859*, 2021.
[194] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Unified pre-training for program understanding and generation," *arXiv preprint arXiv:2103.06333*, 2021.
[195] P. Yin and G. Neubig, "A syntactic neural model for general-purpose code generation," *arXiv preprint arXiv:1704.01696*, 2017.
[196] H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song, "Syntax-directed variational autoencoder for structured data," *arXiv preprint arXiv:1802.08786*, 2018.
[197] M. Allamanis, M. Brockschmidt, and M. Khademi, "Learning to represent programs with graphs," *arXiv preprint arXiv:1711.00740*, 2017.
[198] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," 2022.
[199] S. K. Lahiri, A. Naik, G. Sakkas, P. Choudhury, C. von Veh, M. Musuvathi, J. P. Inala, C. Wang, and J. Gao, "Interactive code generation via test-driven user-intent formalization," 2022.

[200] L. Zhou, J. Gao, D. Li, and H.-Y. Shum, “The design and implementation of xiaoice, an empathetic social chatbot,” *Computational Linguistics*, vol. 46, no. 1, pp. 53–93, 2020.
[201] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, *et al.*, “Towards a human-like open-domain chatbot,” *arXiv preprint arXiv:2001.09977*, 2020.
[202] K. Shuster, J. Xu, M. Komeili, D. Ju, E. M. Smith, S. Roller, M. Ung, M. Chen, K. Arora, J. Lane, *et al.*, “Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage,” *arXiv preprint arXiv:2208.03188*, 2022.
[203] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint arXiv:2005.00341*, 2020.
[204] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, *et al.*, “Codexglue: A machine learning benchmark dataset for code understanding and generation,” *arXiv preprint arXiv:2102.04664*, 2021.
[205] L. Tunstall, L. Von Werra, and T. Wolf, *Natural language processing with transformers.* " O’Reilly Media, Inc.", 2022.
[206] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, “Evaluating Large Language Models Trained on Code,” July 2021. arXiv:2107.03374 [cs].
[207] 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,” *arXiv preprint arXiv:2206.14858*, 2022.
[208] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R Ruiz, J. Schrittwieser, G. Swirszcz, *et al.*, “Discovering faster matrix multiplication algorithms with reinforcement learning,” *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.
[209] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, “Palm: Scaling language modeling with pathways,” *arXiv preprint arXiv:2204.02311*, 2022.
[210] 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*, pp. 248–255, Ieee, 2009.
[211] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing,” *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.
[212] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He, *et al.*, “A comprehensive survey on pretrained foundation models: A history from bert to chatgpt,” *arXiv preprint arXiv:2302.09419*, 2023.
[213] 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.
[214] 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.
[215] A. Borji, “A categorical archive of chatgpt failures,” 2023.
[216] “Could chatgpt become a monster misinformation superspreader?,” Jan 2023.
[217] A. Mahadevan, “This newspaper doesn’t exist: How chatgpt can launch fake news sites in minutes,” Feb 2023.
[218] “Why can’t ai draw realistic human hands?,” Jan 2023.
[219] S. Wang, D. Lo, and L. Jiang, “An empirical study on developer interactions in stackoverflow,” in *Proceedings of the 28th annual ACM symposium on applied computing*, pp. 1019–1024, 2013.
[220] H. Zhang, Y. Li, F. Ma, J. Gao, and L. Su, “Texttruth: an unsupervised approach to discover trustworthy information from multi-sourced text data,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2729–2737, 2018.
[221] B. Goodrich, V. Rao, P. J. Liu, and M. Saleh, “Assessing the factual accuracy of generated text,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 166–175, 2019.
[222] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, pp. 74–81, 2004.
[223] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.
[224] O. Evans, O. Cotton-Barratt, L. Finnveden, A. Bales, A. Balwit, P. Wills, L. Righetti, and W. Saunders, “Truthful ai: Developing and governing ai that does not lie,” *arXiv preprint arXiv:2110.06674*, 2021.
[225] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, *et al.*, “Webgpt: Browser-assisted question-answering with human feedback,” *arXiv preprint arXiv:2112.09332*, 2021.
[226] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli, “Eli5: Long form question answering,” *arXiv preprint arXiv:1907.09190*, 2019.

[227] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," *arXiv preprint arXiv:2109.07958*, 2021.

[228] N. Lee, W. Ping, P. Xu, M. Patwary, M. Shoeybi, and B. Catanzaro, "Factuality enhanced language models for open-ended text generation," *arXiv preprint arXiv:2206.04624*, 2022.

[229] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?," in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 610–623, 2021.

[230] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[231] Z. Kenton, T. Everitt, L. Weidinger, I. Gabriel, V. Mikulik, and G. Irving, "Alignment of language agents," *arXiv preprint arXiv:2103.14659*, 2021.

[232] J. Dhamala, T. Sun, V. Kumar, S. Krishna, Y. Pruksachatkun, K.-W. Chang, and R. Gupta, "Bold: Dataset and metrics for measuring biases in open-ended language generation," in *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 862–872, 2021.

[233] 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*, pp. 6565–6576, PMLR, 2021.

[234] M. Nadeem, A. Bethke, and S. Reddy, "Stereoset: Measuring stereotypical bias in pretrained language models," *arXiv preprint arXiv:2004.09456*, 2020.

[235] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, *et al.*, "Release strategies and the social impacts of language models," *arXiv preprint arXiv:1908.09203*, 2019.

[236] 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.

[237] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer, K. Ndousse, *et al.*, "Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned," *arXiv preprint arXiv:2209.07858*, 2022.

[238] Y. Wu, N. Yu, Z. Li, M. Backes, and Y. Zhang, "Membership inference attacks against text-to-image generation models," *arXiv preprint arXiv:2210.00968*, 2022.

[239] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, "Membership inference attacks on machine learning: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–37, 2022.

[240] J. Duan, F. Kong, S. Wang, X. Shi, and K. Xu, "Are diffusion models vulnerable to membership inference attacks?," *arXiv preprint arXiv:2302.01316*, 2023.

[241] H. Hu and J. Pang, "Membership inference of diffusion models," *arXiv preprint arXiv:2301.09956*, 2023.

[242] T. Matsumoto, T. Miura, and N. Yanai, "Membership inference attacks against diffusion models," *arXiv preprint arXiv:2302.03262*, 2023.

[243] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramèr, and C. Zhang, "Quantifying memorization across neural language models," *arXiv preprint arXiv:2202.07646*, 2022.

[244] N. Carlini, Y. Liu, H. Daume III, U. Erlingsson, T. Kohno, and D. Song, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[245] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[246] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein, "Diffusion art or digital forgery? investigating data replication in diffusion models," *arXiv preprint arXiv:2212.03860*, 2021.

[247] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramèr, B. Balle, D. Ippolito, and E. Wallace, "Extracting training data from diffusion models," *arXiv preprint arXiv:2301.13188*, 2023.

[248] S. Reddy, S. Allan, S. Coghlan, and P. Cooper, "A governance model for the application of ai in health care," *Journal of the American Medical Informatics Association*, vol. 27, no. 3, pp. 491–497, 2020.

[249] Y. Qi and J. Xiao, "Fintech: Ai powers financial services to improve people's lives," *Communications of the ACM*, vol. 61, no. 11, pp. 65–69, 2018.

[250] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[251] Y. Gil, M. Greaves, J. Hendler, and H. Hirsh, "Amplify scientific discovery with artificial intelligence," *Science*, vol. 346, no. 6206, pp. 171–172, 2014.

[252] 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.

[253] O. Ostapenko, T. Lesort, P. Rodríguez, M. R. Arefin, A. Douillard, I. Rish, and L. Charlin, "Continual learning with foundation models: An empirical study of latent replay," in *Conference on Lifelong Learning Agents*, pp. 60–91, PMLR, 2022.

[254] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," *arXiv preprint arXiv:2004.10964*, 2020.

[255] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "Legal-bert: The muppets straight out of law school," *arXiv preprint arXiv:2010.02559*, 2020.

[256] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *arXiv preprint arXiv:2201.11903*, 2022.

[257] Z. Zhang, A. Zhang, M. Li, H. Zhao, G. Karypis, and A. Smola, "Multimodal chain-of-thought reasoning in language models," *arXiv preprint arXiv:2302.00923*, 2023.

[258] A. Madaan, S. Zhou, U. Alon, Y. Yang, and G. Neubig, "Language models of code are few-shot commonsense learners," *arXiv preprint arXiv:2210.07128*, 2022.

[259] 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.

[260] A. Aghajanyan, L. Yu, A. Conneau, W.-N. Hsu, K. Hambardzumyan, S. Zhang, S. Roller, N. Goyal, O. Levy, and L. Zettlemoyer, "Scaling laws for generative mixed-modal language models," *arXiv preprint arXiv:2301.03728*, 2023.

[261] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[262] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[263] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," *Advances in neural information processing systems*, vol. 28, 2015.

[264] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[265] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv*, 2018.

[266] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, "Ernie: Enhanced representation through knowledge integration," *arXiv preprint arXiv:1904.09223*, 2019.

[267] 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.

[268] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," *Advances in neural information processing systems*, vol. 32, 2019.

[269] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mass: Masked sequence to sequence pre-training for language generation," *arXiv preprint arXiv:1905.02450*, 2019.

[270] W. Wang, B. Bi, M. Yan, C. Wu, Z. Bao, J. Xia, L. Peng, and L. Si, "Structbert: Incorporating language structures into pre-training for deep language understanding," *arXiv*, 2019.

[271] M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," *arXiv preprint arXiv:1909.04164*, 2019.

[272] W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju, "Fastbert: a self-distilling bert with adaptive inference time," *arXiv preprint arXiv:2004.02178*, 2020.

[273] 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.

[274] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[275] 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.

[276] 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.

[277] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[278] J. T. Rolfe, "Discrete variational autoencoders," *arXiv preprint arXiv:1609.02200*, 2016.

[279] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," in *International Conference on Machine Learning*, pp. 5583–5594, PMLR, 2021.

[280] 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.

[281] 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.

[282] 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.

[283] 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.

A CURATED ADVANCES IN GENERATIVE AI

In this section, we present a comprehensive review of the recent significant advancements in the field of generative AI. Consistent with the aforementioned discussion, we classify the general framework into unimodal and multimodal generative models. In each subsection, we further categorize the models based on specific modalities, and we provide a table summary of relative paper details.

A.1 Language

In this section, we give a summary of the main milestone models in NLP. Generally, the architecture includes probabilistic objectives, encoder, decoder, and encoder-decoder structure. We also summarize the backbones of these methods.

Year	Conference	Method	Architecture	Backbone	Code
2013	NeurIPS	Skip-Gram [261]	Probabilistic	Word2Vec	https://github.com/.../models
2014	EMNLP	GloVe [262]	Probabilistic	Word2Vec	https://github.com/.../GloVe
2015	NeurIPS	LM-LSTM [263]	Probabilistic	LSTM	-
2017	TACL	FastText [264]	Probabilistic	Word2Vec	https://github.com/.../fastText
2018	NAACL	ELMO [265]	Encoder	LSTM	https://allennlp.org/elmo
2018	-	GPT [61]	Decoder	Transformer	https://github.com/huggingface/transformers
2019	ACL	ERNIE [266]	Encoder	Transformer	https://github.com/.../ERNIE
2019	ACL	Transformer-XL [267]	Encoder	Transformer	https://github.com/.../transformer-xl
2019	NeurIPS	UNILM [268]	Encoder	Transformer	https://github.com/.../unilm
2019	NAACL	BERT [42]	Encoder	Transformer	https://github.com/google-research/bert
2019	CoRR	RoBERTa [43]	Encoder	Transformer	https://github.com/pytorch/fairseq
2019	NeurIPS	XLNet [44]	Encoder	Transformer	https://github.com/zihangdai/xlnet
2019	NeurIPS	DistilBERT [214]	Encoder	Transformer	https://github.com/huggingface/transformers
2019	ICML	MASS [269]	Encoder	Transformer	https://github.com/microsoft/MASS
2019	ICLR	StructBERT [270]	Encoder	Transformer	-
2019	EMNLP	KnowBERT [271]	Encoder	Transformer	https://github.com/.../kb
2019	-	GPT-2 [245]	Decoder	Transformer	https://github.com/openai/gpt-2
2019	JMLR	T5 [56]	Encoder-Decoder	Transformer	https://github.com/google-research/text-to-text-transfer-transformer
2019	-	Megatron [65]	General	Transformer	https://github.com/NVIDIA/Megatron-LM
2020	ACL	fastBERT [272]	Encoder	Transformer	https://github.com/.../FastBERT
2020	ACL	spanBERT [273]	Encoder	Transformer	https://github.com/.../SpanBERT
2020	ICLR	Reformer [274]	Encoder	Reformer	https://github.com/.../reformer
2020	EMNLP	TinyBERT [275]	Encoder	Transformer	https://github.com/.../TinyBERT
2020	ICLR	ALBERT [276]	Encoder	Transformer	https://github.com/google-research/ALBERT
2020	ICLR	ELECTRA [213]	Encoder	Transformer	https://github.com/google-research/electra
2020	NeurIPS	GPT-3 [245]	Decoder	Transformer	https://github.com/openai/gpt-3
2020	ACL	BART [34]	Encoder-Decoder	Transformer	https://github.com/huggingface/transformers
2020	-	PaLM [209]	Decoder	Transformer	https://github.com/lucidrains/PaLM-pytorch
2021	-	Gopher [39]	Decoder	Transformer	-
2021	JMLR	Switch [67]	Encoder-Decoder	Transformer	https://github.com/tensorflow/mesh
2022	-	LaMDA [236]	Decoder	Transformer	https://github.com/.../LaMDA
2022	-	OPT [45]	Decoder	Transformer	https://github.com/facebookresearch/metaseq
2022	-	InstructGPT [10]	Decoder	Transformer	https://github.com/openai/following-instructions-human-feedback
2022	-	Sparrow [46]	Decoder	Transformer	-
2022	-	BLOOM [64]	Decoder	Transformer	https://github.com/bigscience-workshop/bigscience
2022	-	MT-NLG [66]	Decoder	Transformer	https://github.com/microsoft/DeepSpeed
2022	ICLR	HTLM [69]	Encoder-Decoder	Transformer	-
2022	ACL	DQ-BART [70]	Encoder-Decoder	Transformer	https://github.com/amazon-research/dq-bart
2022	ICLR	ExT5 [68]	Encoder-Decoder	Transformer	https://github.com/google-research/text-to-text-transfer-transformer
2023	-	LLaMA [277]	Decoder	Transformer	-

Table 2. Major natural language models.

A.2 Vision

Year	Method	Architecture	Category	Code
2014	GAN [29]	GAN	Traditional	https://github.com/goodfeli/adversarial
2015	LAPGAN [71]	GAN	Traditioal	https://github.com/facebook/eyescream
2015	DCGANs [73]	GAN	Traditioal	https://github.com/carpedm20/DCGAN
2017	Progressive GAN [74]	GAN	Traditioal	https://github.com/tkarras/progre...
2019	SAGAN [75]	GAN	Traditioal	https://github.com/brain-research/self
2018	BigGAN [76]	GAN	Traditioal	https://github.com/ajbrock/BigGAN
2019	StyleGAN [77]	GAN	Traditioal	https://github.com/NVlabs/stylegan
2016	BiGAN [78]	GAN	Traditioal	https://github.com/eriklindernoren/GAN
2018	AGE [79]	GAN	Traditioal	https://github.com/DmitryUlyanov/AGE
2017	D2GAN [80]	GAN	Traditioal	https://github.com/tund/D2GAN
2016	GMAN [81]	GAN	Traditioal	https://github.com/zhengchuanpan/GMAN
2017	MGAN [82]	GAN	Traditioal	https://github.com/qhoangdl/MGAN
2018	MAD-GAN [83]	GAN	Traditioal	https://github.com/LiDan456/MAD-GANs
2016	CoGAN [84]	GAN	Traditioal	https://github.com/mingyuliutw/CoGAN
2016	InfoGAN [85]	GAN	Representative variants	https://github.com/eriklindernoren/...GAN
2014	CGANs [86]	GAN	Representative variants	https://github.com/pfnet-research/sngan...
2018	C-CycleGAN [87]	GAN	Representative variants	-
2019	MSGAN [88]	GAN	Representative variants	https://github.com/HelenMao/MSGAN
2016	f-GAN [89]	GAN	Representative variants	https://github.com/mboudiaf/Mut...
2017	WGAN [90]	GAN	Objective	https://github.com/daheyinyin/wgan
2020	GLS-GAN [91]	GAN	Objective	https://github.com/guojunq/lsgan
2017	LS-GAN [92]	GAN	Objective	https://github.com/xudonmao/LSGAN
2018	SNGAN [93]	GAN	Objective	https://github.com/pfnet-research/sngan
2016	Che et al. [94]	GAN	Objective	-
2016	UnrolledGAN [95]	GAN	Objective	https://github.com/poolio/unrolledgan
2018	RelativisticGAN [96]	GAN	Objective	https://github.com/AlexiaJM/RelativisticGAN
2013	VAE [30]	VAE	Traditional	https://github.com/AntixK/PyTorch-VAE
2018	VampPrior [99]	VAE	Complex priors	https://github.com/jmtomczak/vaevampprior
2019	BIVA[100]	VAE	Complex priors	https://github.com/vlievin/biva-pytorch
2020	NVAE[101]	VAE	Complex priors	https://github.com/NVlabs/NVAE
2021	GHVAE [102]	VAE	Complex priors	-
2019	RAE [103]	VAE	Regularized Autoencoders	https://github.com/ParthaEth/Regul...
2021	dGAN [278]	VAE	Regularized Autoencoders	https://github.com/topics/discrete...
2017	VQ-VAE [104]	VAE	Regularized Autoencoders	https://github.com/deepmind/...vqvae
2019	VQ-VAE2 [105]	VAE	Regularized Autoencoders	https://github.com/deepmind/sonnet
2014	NICE [57]	Flow	Coupling and autoregressive	https://github.com/EugenHotaj/...nice
2016	Real NVP [106]	Flow	Coupling and autoregressive	https://github.com/tensorflow/models
2017	MAF [107]	Flow	Coupling and autoregressive	https://github.com/gpapamak/maf
2018	NAF [108]	Flow	Coupling and autoregressive	https://github.com/CW-Huang/NAF
2020	BNAF [109]	Flow	Coupling and autoregressive	https://github.com/nicola-decao/BNAF
2017	ConvFlow [110]	Flow	Convolutional and Residual	-
2019	E-ConvFlow [111]	Flow	Convolutional and Residual	https://github.com/ehoogeboom/emerging
2017	RevNets [112]	Flow	Convolutional and Residual	https://github.com/renmengye/revnet-public
2018	i-RevNets [113]	Flow	Convolutional and Residual	https://github.com/jhjacobsen/pytorch-i-revnet
2020	DDPM [115]	Diffusion	Traditional	https://github.com/hojonathanho/diffusion
2019	NCSN [31]	Diffusion	Traditional	https://github.com/ermongroup/ncsn
2020	Score SDE [116]	Diffusion	Training Enhance	https://github.com/yang-song/scoresde
2022	Salimans et al. [117]	Diffusion	Training Enhance	https://github.com/Hramchenko/diffusion..distiller
2022	TDPM [118]	Diffusion	Training Enhance	https://github.com/jegzheng/truncat..
2022	ES-DDPM [119]	Diffusion	Training Enhance	https://github.com/zhaoyanglyu/early...
2022	Franzese et al. [120]	Diffusion	Training Enhance	-
2021	Improved DDPM [121]	Diffusion	Training Enhance	https://github.com/openai/improved-diffusion
2021	San Roman et al. [122]	Diffusion	Training Enhance	-
2020	DDIM [123]	Diffusion	Training-free Sampling	https://github.com/ermongroup/ddim
2022	Analytic-DPM [124]	Diffusion	Training-free Sampling	https://github.com/baofff/Analytic-DPM
2021	Watson et al. [125]	Diffusion	Training-free Sampling	-
2022	Watson et al. [126]	Diffusion	Training-free Sampling	-
2021	Nachmani et al. [127]	Diffusion	Noise Distribution	-
2022	Cold Diffusion [128]	Diffusion	Noise Distribution	https://github.com/arpitbansal297/cold-...
2021	CCDF [129]	Diffusion	Noise Distribution	-
2022	DiffuseVAE [130]	Diffusion	Mixed Modeling	https://github.com/kpandey008/DiffuseVAE
2021	LSGM [131]	Diffusion	Mixed Modeling	https://github.com/NVlabs/LSGM
2021	Denoising diffusion GANs [132]	Diffusion	Mixed Modeling	https://github.com/NVlabs/denoising
2021	DiffFlow [133]	Diffusion	Mixed Modeling	https://github.com/qsh-zh/DiffFlow

Table 3. Major vision generative models.

A.3 Vision Language

Year	Method	Task	Architecture	Code
2019	VisualBERT [134]	VL Encoders	Concatenated Encoders	https://github.com/uclanlp/visualbert
2020	VL-BERT [136]	VL Encoders	Concatenated Encoders	https://github.com/jackroos/VL-BERT
2020	UNITER [138]	VL Encoders	Concatenated Encoders	https://github.com/ChenRocks/UNITER
2021	ViLT [279]	VL Encoders	Concatenated Encoders	https://github.com/dandelin/vilt
2022	SimVLM [135]	VL Encoders	Concatenated Encoders	-
2019	LXMERT [139]	VL Encoders	Cross-aligned Encoders	https://github.com/airsplay/lxmert
2019	X-LXMERT [280]	VL Encoders	Cross-aligned Encoders	https://github.com/allenai/x-lxmert
2020	PixelBERT [281]	VL Encoders	Concatenated Encoders	https://github.com/microsoft/xpretrain
2019	ViLBERT [140]	VL Encoders	Cross-aligned Encoders	-
2021	WenLan [282]	VL Encoders	Cross-aligned Encoders	https://github.com/BAAI-WuDao/BriVl
2021	CLIP [245]	VL Encoders	Cross-aligned Encoders	https://github.com/openai/CLIP
2019	VLP [138]	To-text Decoders	Encoder-Decoders	https://github.com/LuoweiZhou/VLP
2021	ALBEF [141]	To-text Decoders	Encoder-Decoders	https://github.com/salesforce/ALBEF
2022	BLIP [142]	To-text Decoders	Encoder-Decoders	https://github.com/salesforce/lavis
2023	BLIP-2 [283]	To-text Decoders	Frozen Decoders	https://github.com/salesforce/lavis
2021	Frozen [143]	To-text Decoders	Frozen Decoders	-
2022	Flamingo [144]	To-text Decoders	Frozen Decoders	https://github.com/lucidrains/flamingo-pytorch
2023	Grounding [145]	To-text Decoders	Frozen Decoders	https://github.com/kohjingyu/fromage
2017	StackGAN [148]	To-image Decoders	GAN Decoders	https://github.com/hanzhanggit/StackGAN
2018	AttnGAN [149]	To-image Decoders	GAN Decoders	https://github.com/taoxugit/AttnGAN
2021	StyleCLIP [150]	To-image Decoders	GAN Decoders	https://github.com/orpatashnik/StyleCLIP
2021	GLIDE [151]	To-image Decoders	Diffusion Decoders	https://github.com/openai/glide-text2im
2022	Stable-diffusion [13]	To-image Decoders	Diffusion Decoders	https://github.com/compvis/stable-diffusion
2022	Imagen [152]	To-image Decoders	Diffusion Decoders	https://github.com/lucidrains/imagen-pytorch
2022	DALL-E-2 [5]	To-image Decoders	Diffusion Decoders	https://github.com/lucidrains/DALLE2-pytorch
2021	DALL-E [1]	To-image Decoders	VAE Decoders	https://github.com/openai/DALL-E

Table 4. Major vision language models.

A.4 Text Audio

Year	Method	Task	Code
2021	AdaSpeech [153]	Text-Audio Generation	https://github.com/rishikksh20/AdaSpeech
2021	AdaSpeech2	Text-Audio Generation	https://github.com/rishikksh20/AdaSpeech2
2020	Lombard [154]	Text-Audio Generation	https://github.com/dipjyoti92/TTS-Style-Transfer
2019	Zhang et al.[156]	Text-Audio Generation	https://github.com/PaddlePaddle/PaddleSpeech
2019	Yu et al.[157]	Text-Music Generation	-
2018	JTAV [158]	Text-Music Generation	https://github.com/mengshor/JTAV
2021	Ferraro et al.[159]	Text-Music Generation	https://github.com/andrebola/contrastive-mir-learning
2016	Choi et al.[160]	Text-Music Generation	-
2021	MusCaps [161]	Text-Music Generation	https://github.com/ilaria-manco/muscaps
2022	Manco et al.[162]	Text-Music Generation	https://github.com/ilaria-manco/mulap
2022	CLAP [163]	Text-Music Generation	https://github.com/YuanGongND/vocalsound
2020	Jukebox [203]	Text-Music Generation	https://github.com/openai/jukebox

Table 5. Major text audio models.

A.5 Text Graph

Year	Method	Task	Code
2016	Li et al. [165]	Text-to-KG Generation	-
2019	KG-BERT [166]	Text-to-KG Generation	https://github.com/yao8839836/kg-bert
2020	Malaviya et al. [167]	Text-to-KG Generation	https://github.com/allenai/commonsense-kg-completion
2019	Petroni et al. citepetroni2019language	Text-to-KG Generation	https://github.com/facebookresearch/LAMA
2020	Shin et al. citeshin2020autoprompt	Text-to-KG Generation	https://github.com/ucinlp/autoprompt
2021	Li et al. citeli2021prefix	Text-to-KG Generation	https://github.com/XiangLi1999/PrefixTuning
2022	Lu et al. [173]	Text-to-KG Generation	https://github.com/universal-ie/UIE
2022	Grapher [174]	Text-to-KG Generation	https://github.com/ibm/grapher
2020	CycleGT [171]	Text-KG Generation	https://github.com/QipengGuo/CycleGT
2020	DualTKB [172]	Text-KG Generation	-
2018	GTR-LSTM [176]	KG-to-Text Generation	-
2018	Song et al. [177]	KG-to-Text Generation	https://github.com/freesunshine0316/neural-graph-to-seq-mp
2020	DUALENC [175]	KG-to-Text Generation	https://github.com/zhaochaocs/DualEnc
2019	Koncel-Kedziorski et al. [178]	KG-to-Text Generation	https://github.com/rikdz/GraphWriter
2020	Ribeiro et al. [180]	KG-to-Text Generation	https://github.com/UKPLab/kg2text
2020	HetGT [181]	KG-to-Text Generation	https://github.com/QAQ-v/HetGT
2016	Dong et al. [182]	Semantic Parsing	https://github.com/donglixp/lang2logic
2016	Jia et al. [183]	Semantic Parsing	https://worksheets.codalab.org/...
2018	Lyu et al. [184]	Semantic Parsing	https://github.com/...PREDICTION
2018	Chen et al. [185]	Semantic Parsing	https://github.com/dongpobeyond/Seq2Act
2019	Zhang et al. [186]	Semantic Parsing	https://github.com/sheng-z/stog
2019	Fancellu et al. [187]	Semantic Parsing	-
2021	Text2Mol [189]	Text-Molecule Generation	https://github.com/cnedwards/text2mol
2022	MolT5 [190]	Text-Molecule Generation	https://github.com/blender-nlp/MolT5
2022	MoMu [188]	Text-Molecule Generation	https://github.com/bingsu12/momu

Table 6. Major text graph models.

A.6 Text Code

Year	Method	Task	Code
2020	CodeBERT [191]	Text-Code Generation	https://github.com/microsoft/CodeBERT
2020	CodeBERT [191]	Text-Code Generation	https://github.com/microsoft/CodeBERT
2020	CuBERT [192]	Text-Code Generation	https://github.com/google-research/google-research/tree/master/cubert
2021	CodeT5 [193]	Text-Code Generation	https://github.com/salesforce/codet5
2021	PLBART [194]	Text-Code Generation	https://github.com/wasiahmad/PLBART
2017	Yin et al. [195]	Text-Code Generation	https://github.com/pcyin/NL2code
2018	Dai et al. [196]	Text-Code Generation	https://github.com/Hanjun-Dai/sdvae
2022	CODEGEN [198]	Text-Code Generation	https://github.com/salesforce/CodeGen
2022	TDUIF [199]	Text-Code Generation	-

Table 7. Major text code models.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009