NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

THEORETICAL INFORMATICS

MSc THESIS

Generative AI in Graphics

Evgenios N. Mazarakis

Supervisors: **Theocharis Theocharis**, Professor NKUA
TODO , Professor NKUA

ATHENS

DECEMBER 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΘΕΩΡΗΤΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Generative AI in Graphics

Ευγένιος Ν. Μαζαράκης

Επιβλέποντες: **Θεοχάρης Θεοχάρης**, Καθηγητής ΕΚΠΑ
TODO , Καθηγητής ΕΚΠΑ

ΑΘΗΜΑ

ΔΕΚΕΜΒΡΙΟΣ 2023

MSc THESIS

Generative AI in Graphics

Evgenios N. Mazarakis
S.N.: M1458

SUPERVISORS: **Theocharis Theocharis**, Professor NKUA
TODO , Professor NKUA

EXAMINATION COMMITTEE: **TODO** , Professor NKUA
TODO , Professor NKUA
TODO , Professor NKUA

Examination Date: **DD Month Year**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Generative AI in Graphics

Ευγένιος Ν. Μαζαράκης
Α.Μ.: M1458

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Θεοχάρης Θεοχάρης**, Καθηγητής ΕΚΠΑ
TODO , Καθηγητής ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **TODO** , Καθηγητής ΕΚΠΑ
TODO , Καθηγητής ΕΚΠΑ
TODO , Καθηγητής ΕΚΠΑ

Ημερομηνία Εξέτασης: **DD Month Year**

ABSTRACT

This master's thesis delves into the cutting-edge domain of Generative Artificial Intelligence (AI) with a primary focus on graphics applications. The study explores various deep learning techniques, including Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Diffusion Models, and Transformers. These methods have revolutionized the field by enabling the generation of realistic and high-quality content, ranging from images to entire scenes.

The investigation begins with an in-depth exploration of Variational Autoencoders, emphasizing their role in capturing latent representations of data. Subsequently, the research transitions to Generative Adversarial Networks, discussing their adversarial training process for generating authentic content. Additionally, the study reviews Diffusion Models, which excel in probabilistic generative modeling, and Transformers, renowned for their success in sequential data generation tasks.

A significant portion of the thesis is dedicated to the execution of four models in the graphics domain based on some parameters. Executing generative models through **Python code** empowers users to harness advanced AI techniques. This enables the creation of diverse, high-quality content, revolutionizing applications in various domains seamlessly.

The findings highlight the transformative impact of these generative models on the field of graphics, showcasing their ability to create immersive and novel visual content through the amalgamation of sophisticated deep learning techniques.

SUBJECT AREA: Generative AI

KEYWORDS: Generative AI, Artificial Intelligence, Models, Graphics, TODO

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία εμβαθύνει στον τομέα αιχμής της Generative Artificial Intelligence (AI) εστιάζοντας στις εφαρμογές των γραφικών. Η μελέτη διερευνά διάφορες deep learning τεχνικές, συμπεριλαμβανομένων των Variational Autoencoders (VAEs), των Generative Adversarial Networks (GANs), των Diffusion Models και των Transformers. Αυτές οι μέθοδοι έχουν φέρει επανάσταση στον τομέα της Τεχνητής Νοημοσύνης, επιτρέποντας τη δημιουργία ρεαλιστικού και υψηλής ποιότητας περιεχομένου, που κυμαίνεται από εικόνες έως ολόκληρες σκηνές.

Η διπλωματική εργασία ξεκινά με μια εις βάθος εξερεύνηση των Variational Autoencoder, δίνοντας έμφαση στον ρόλο τους στην λανθάνουσα αναπαράσταση των δεδομένων. Στη συνέχεια, η έρευνα μεταβαίνει στα Generative Adversarial Networks, αναδεικνύοντας τη διαδικασία εκπαίδευσης των συστατικών τους στοιχείων για τη δημιουργία αυθεντικού περιεχομένου. Επιπλέον, η μελέτη εξετάζει τα Diffusion Models, τα οποία διαπρέπουν στην probabilistic generative μοντελοποίηση, και τους Transformers , γνωστούς για την επιτυχία τους σε εργασίες παραγωγής διαδοχικών δεδομένων (sequential data).

Ένα σημαντικό μέρος της διπλωματικής εργασίας είναι αφιερωμένο στην εκτέλεση τεσσάρων μοντέλων στον τομέα των γραφικών με βάση ορισμένες παραμέτρους. Η εκτέλεση των μοντέλων μέσω **κώδικα Python** δίνει τη δυνατότητα στους χρήστες να αξιοποιήσουν προηγμένες τεχνικές AI. Αυτό επιτρέπει τη δημιουργία ποικίλου περιεχομένου υψηλής ποιότητας, φέρνοντας επανάσταση στις εφαρμογές σε διάφορους τομείς απρόσκοπτα.

Τα ευρήματα υπογραμμίζουν την επίδραση αυτών των παραγωγικών μοντέλων στο πεδίο των γραφικών, επιδεικνύοντας την ικανότητά τους να δημιουργούν πρωτότυπο και νέο οπτικό περιεχόμενο μέσω της συγχώνευσης εξελιγμένων τεχνικών βαθιάς μάθησης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Γενεσιουργός Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΙΕΔΙΑ: Γενεσιουργός Τεχνητή Νοημοσύνη, Τεχνητή Νοημοσύνη, Μοντέλα, Γραφικά, TODO

Η διπλωματική εργασία
είναι αφιερωμένη
στο Νίκο Μαζαράκη,
στην Έλλη Μαζαράκη,
και στο Θεόφιλο Μαζαράκη.

ACKNOWELEDGMENTS

I feel the need to express my immense gratitude to my parents Nico and Elli Mazarakis and to my beloved brother Theofilos Mazarakis, for everything they have offered me during my student years as well for their undivided support in my every choice. At this point, I would like to thank my supervisor Mr. Theoharis Theoharis once again for trusting and devoting valuable time to get the job done.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

The aim of this thesis is to focus on Generative AI algorithms, in order to list and classify the available Generative AI algorithms and applications that can be exploited in the scientific fields of Graphics.

We first provide all the necessary definitions commonly used around the AI, Generative AI, Generative AI model. This is the necessary background we need to be able to understand what is mentioned in this thesis. Then we talk about some fundamental deep learning architectures that used on generative AI models. Thereafter, we describe the implementation phases of a Generative AI models. Then we offer a classification of some of the advanced GAI algorithms. The categorization of the models have been done based on the generated content, output.

In addition, in the second chapter we examine four models. We run the models, based on some parameters that we defined. **TODO**

Finally, in the third and last chapter of this thesis there are the conclusions obtained from the execution of the Generative AI models for the graphics. It also mentions some important challenges around models in the field of graphics.

ΠΡΟΛΟΓΟΣ

Στόχος της παρούσας διπλωματικής εργασίας είναι να εστιάσει στους αλγορίθμους Generative AI, προκειμένου να παραθέσει και να ταξινομήσει τους διαθέσιμους αλγόριθμους και τις εφαρμογές του Generative AI που μπορούν να αξιοποιηθούν στο επιστημονικό πεδίο των Γραφικών.

Αρχικά παρέχουμε όλους τους απαραίτητους ορισμούς για το AI, Generative AI και Generative AI model. Αυτό είναι το απαραίτητο υπόβαθρο που χρειαζόμαστε για να μπορέσουμε να κατανοήσουμε ότι αναφέρεται σε αυτή την εργασία. Στη συνέχεια μιλάμε για μερικές αρχιτεκτονικές βαθιάς μάθησης (deep learning) που χρησιμοποιούνται από τα Generative AI model. Έπειτα, περιγράφουμε τις φάσεις υλοποίησης ενός Generative AI μοντέλου. Ακόμα προσφέρουμε μια ταξινόμηση ορισμένων από τους προηγμένους αλγόριθμους GAI. Η κατηγοριοποίησή των μοντέλων έχει γίνει με βάση το παραγόμενο περιεχόμενο, την έξοδο, που δημιουργούν.

Επιπλέον, στο δεύτερο κεφάλαιο εξετάζουμε τέσσερα μοντέλα. Ορίζουμε κάποιες παραμέτρους και βάσει αυτών τρέχουμε τα μοντέλα. **TODO**

Τέλος, στο τρίτο και τελευταίο κεφάλαιο της παρούσας εργασίας υπάρχουν τα συμπεράσματα που προέκυψαν από την εκτέλεση των μοντέλων Generative AI για τα γραφικά. Αναφέρει επίσης ορισμένες σημαντικές προκλήσεις σχετικά με τα μοντέλα στον τομέα των γραφικών.

1. INTRODUCTION

The thesis is organized as follows. Chapter 1 contains all the necessary definitions around the area of generative AI and also a taxonomy of some AI models based on the generated content. Chapter 2 introduces some models which we ran based on some parameters. Finally, we mention our conclusions based on the run of the models. The structure of the master thesis is given in the following figure.

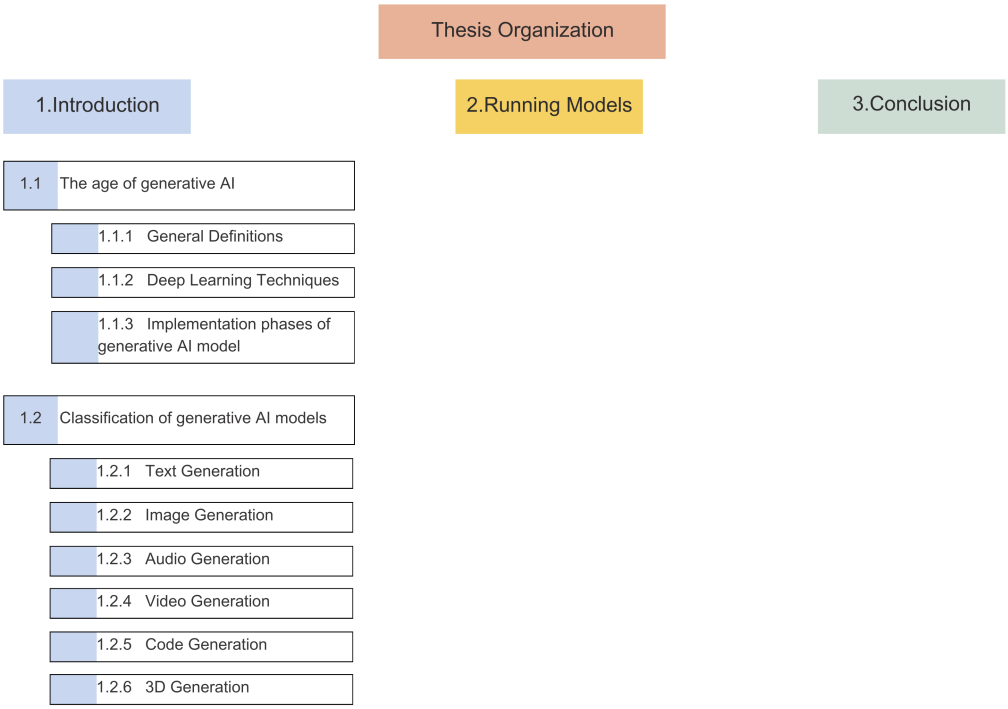

Figure 1.1: Organization of this thesis.

1.1 The age of generative AI

Generative AI is a branch of AI that can create new content such as texts, images, or audio that increasingly often cannot be distinguished any more from human craftsmanship. Generative AI became known almost worldwide on November 30, 2022, when OpenAI released ChatGPT. A chat where the user can ask questions and get detailed answers.ChatGPT's ease of use also for

non-expert users was a core contributing factor to its explosive worldwide adoption.

A new era in the synthesis and manipulation of digital content has begun. This application leverage the prowess of deep learning architecture. Generative AI models, equipped with vast data sets and intricate designs, have the extraordinary capability to create new and diverse content. These models can be trained to generate genuinely different multimedia formats, like video, audio, or text, from various input formats. For instance, generative AI can be used to create realistic images from textual description, produce video content from textual descriptions.

The birth of Generative AI, as we know it today, was heralded by the advent of a type of machine learning known as neural networks. Inspired by the human brain, these models use interconnected layers of “neurons” to process and learn from data. A neural network is trained to recognize patterns in a dataset. Once the network is trained, it can make decisions or predictions without being explicitly programmed to perform tasks.

The high-level view of Generative AI consists of three things, the model, the system and the application. A generative AI model refers to generative modeling that is instantiated with a machine learning architecture (e.g., a deep neural network) and, therefore, can create new data samples based on learned patterns. Further, a generative AI system encompasses the entire infrastructure, including the model, data processing, and user interface components. The model serves as the core component of the system. Lastly, generative AI applications refer to the practical use cases and implementations of these systems, such as code generation that solve real-world problems and drive innovation across various domains. [22]

The current state of Generative AI has some limitations, such as incorrect outputs, bias and fairness, copyright violation, environmental concerns.

The creative power of Generative AI comes from a specific type of neural network called a Generative Adversarial Network (GAN), which was proposed by Ian Goodfellow and his colleagues in 2014 [26]. In 2017, Google researchers develop the concept of transformers in the research paper ”Attention is All You Need” [64]. The Transformer model, a novel architecture that revolutionized the field of natural language processing (NLP) and became the basis for the large language models (LLMs) we now know.

As we look to the future, it’s clear that generative AI will continue to shape our world in ways we can’t yet imagine. Industry reports suggest that generative AI could raise global gross domestic product (GDP) by 7% and replace 300 million jobs of knowledge workers. [22]

1.1.1 General Definitions

Definition 1.1.1 (AI)**.** Artificial Intelligence (AI) refers to systems that display intelligent behaviour by analysing their environment and taking actions – with some degree of autonomy – to achieve specific goals. [62]

Definition 1.1.2 (Generative AI)**.** Generative AI refers to artificial intelligence that can generate novel content, rather than simply analysing or acting on existing data like expert system. [27]

Machine learning serves as an essential foundation for generative AI. Machine learning is a field

of study that emphasizes how to build effective algorithms using data, enabling computers to gain new knowledge from data. It can facilitate generative AI to learn new content from vast amounts of data and create diverse content based on different datasets. Machine learning constitutes a crucial foundation for AI and encompasses discriminative and generative models.

Definition 1.1.3 (ML)**.** A computer program is said to learn from **experience E** with respect to some **class of tasks T**, and **performance measure P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**. [42]

Definition 1.1.4 (Model)**.** A formal description of a system, process or equation used to simplify a complex subject. [57]

Definition 1.1.5 (Discrim. Model)**.** Discriminative models determine a conditional probability to accomplish classification and decision-making given data. [40]

Definition 1.1.6 (Gen. Model)**.** Generative models directly predict a distribution and generate new data. [40]

Definition 1.1.7 (Model Parameters)**.** A model parameter is a configuration variable that is internal to the model and whose value can be estimated from data. They are the part of the model that is learned from historical training data. Parameters is a synonym for weights of neural networks. [25]

Definition 1.1.8 (Model Hyper-parameters)**.** A model hyper-parameter is a configuration that is external to the model and whose value cannot be estimated from data. Hyper-parameters are settings that use to control the behaviour of the learning algorithm/model. [25]

Deep neural networks, which are trained on massive datasets to learn their fundamental patterns and probability distributions, are the backbone of generative AI. These networks then employ generative models to generate new data.

Definition 1.1.9 (DNN)**.** Deep neural networks (DNN) is a class of machine learning algorithms similar to the artificial neural network and aims to mimic the information processing of the brain. DNN shave more than one hidden layer situated between the input and output layers. [25]

Based on the kind of data available and the research question at hand, a scientist will choose to train an algorithm using a specific learning model. There are four specific learning models, Supervised, Unsupervised, Semi-Supervised and Reinforcement learning.

Definition 1.1.10 (Supervised)**.** Supervised learning is a technique for training AI systems, in which a neural network learns to make predictions or classifications based on a training dataset of labelled examples. That means having a full set of labelled data while training an algorithm. Fully labelled means that each example in the training dataset is tagged with the answer the algorithm should come up with on its own.There are two main areas where supervised learning is useful: classification problems and regression problems. [43]

Definition 1.1.11 (Unsupervised)**.** In unsupervised learning a trove of unlabelled data is fed into the neural network, which begins looking for patterns in that data without the help of labels. A deep learning model is handed a dataset without explicit instructions on what to do with it. The training dataset is a collection of examples without a specific desired outcome or correct answer. The neural network then attempts to automatically find structure in the data by extracting useful features and analysing its structure. [43]

Definition 1.1.12 (Semi-Supervised)**.** Semi-supervised learning is a training dataset with both labelled and unlabelled data. This method is particularly useful when extracting relevant features from the data is difficult, and labelling examples is a time-intensive task for experts. [43]

Definition 1.1.13 (Reinforcement)**.** Reinforcement learning is a method for optimizing an AI system by rewarding desirable behaviours and penalizing undesirable ones. The AI system just receives an occasional reward (or punishment) signal in response to the actions that it takes. [43]

Definition 1.1.14 (Foundation Model)**.** A foundation model is any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuned) to a wide range of downstream tasks. They are based on deep neural networks and self-supervised learning. [8]

Definition 1.1.15 (LLM)**.** A large language model (LLM) refers to neural networks for modelling and generating text data that typically combine three characteristics. First, the language model uses a large-scale, sequential neural network (e.g., transformer with an attention mechanism). Second, the neural network is pre-trained through self-supervision in which auxiliary tasks are designed to learn a representation of natural language without risk of over-fitting (e.g., next-word prediction). Third, the pre-training makes use of large-scale datasets of text (e.g., Wikipedia, or even multi-language datasets). [22]

Definition 1.1.16 (Unimodal)**.** Unimodal (Single-modal) models take instructions from the same input type as their output. [22]

Definition 1.1.17 (Multimodal)**.** Multimodal models can take input from different sources and generate output in various forms. [22]

1.1.2 Deep Learning Techniques

The model's architecture determines how it processes and generates information, which makes it a critical aspect of its functionality and suitable for specific tasks. These architectural choices have significant implications for how the models generate new data points and learn from the available data. Among the most popular generative AI model's architecture are Generative Adversarial Networks (GANs), Variational Auto-encoders (VAEs), Diffusion and Transformer-based models. In this section we will do a brief introduction to these well know techniques.

Variational Auto-Encoders (VAEs)

Dimensionality Reduction
In machine learning, dimensionality reduction is the process of reducing the number of features that describe some data. The dimensionality reduction can be made in two different ways: by selection where only some existing features are conserved or by extraction where a reduced number of new features are created based on the old features and containing basically the same information as the input data. [60]

First, let's call encoder the process that produce the "new features" representation from the "old features" representation (by selection or by extraction) and decoder the reverse process. Dimensionality reduction can then be interpreted as data compression where the encoder compresses the data (from the initial space to the encoded space, also called latent space) whereas

the decoder decompresses them.The main purpose of a dimensionality reduction method is to find the best encoder/decoder pair among a given family. In other words, for a given set of possible encoders and decoders, we are looking for the pair that keeps the maximum of information when encoding and, so, has the minimum of reconstruction error when decoding. If we denote respectively E and D the families of encoders and decoders we are considering, then the dimensionality reduction problem can be written as:

$$(e^*, d^*) = \underset{(e,d)\in E\times D}{\arg\min}\ \epsilon(x, d(e(x)))$$

A notation for the above equation:

1. (e^*, d^*): the best encoder/decoder pair among a given family
2. $\epsilon(x, d(e(x)))$: defines the reconstruction error measure between the input data x and the output (encoded-decoded) data d(e(x)). [33]

Understanding Auto-Encoders
Before diving into VAEs, let's briefly review the concept of autoencoders. An autoencoder is an unsupervised learning algorithm that aims to learn compressed representation, or encoding, of the input data. Unsupervised learning can be thought of as a form of independent exploration or discovery. In this approach, an algorithm is given a set of input data without any explicit labels. The goal of unsupervised learning is for the algorithm to uncover hidden patterns, structures, or relationships within the data on its own. The general idea of autoencoders is pretty simple and consists in setting an encoder and a decoder as neural networks and to learn the best encoding-decoding scheme using an iterative optimisation process. So, at each iteration we feed the autoencoder architecture with some data, we compare the encoded-decoded output with the initial data and backpropagate the error through the architecture to update the weights of the networks. autoencoders consist of an Encoder network that maps the input data to a lower-dimensional latent space and a Decoder network that reconstructs the original data from the latent representation. The Encoder and Decoder are trained together to minimize the reconstruction error, encouraging the autoencoder to capture the most salient features of the input data. [33]

Introducing Variational Auto-Encoders
The Variational Autoencoder (VAE) [35] was first introduced by Diederik P. Kingma et al. in 2013. This model provides a concise way of capturing the essential low-dimensional information from the data, which can be used to generate new samples through simple manipulation of the learned low-dimensional representations via a decoder. VAEs is composed of an encoder and a decoder. The encoder extracts the mean and variance of the latent variables that determine its properties from the data through a neural network. The decoder adds Gaussian noise to the encoded information to generate new data. VAEs force the latent variables of the latent space to a standard normal distribution. The encoder is no longer given a point but a distribution, allowing the model to learn a smooth latent state representation of the input data. This enables the model to cover unseen samples in the input data.(see figure 1.2, page 17) [39]

Key Components of VAEs [35], [34]

Encoder
In the world of VAEs, the Encoder acts as an interpreter who not only comprehend the substance of the input data moreover captures the complexity and delicate details of the data. It maps the data point to a probability distribution that spans the latent space. This distribution unveiling the potential ways the data point can be interpreted within the latent space.

Latent Space
The latent space is a lower-dimensional representation that captures the essence and core elements of the input data. The latent space should have good properties that enable generative process. In order to be able to use the decoder of our variational autoencoder for generative purpose, we have to be sure that the latent space is regular enough. The regularity that is expected from the latent space in order to make generative process possible can be expressed through two main properties: continuity that is two close points in the latent space should not give two completely different contents once decoded and completeness that is for a chosen distribution, a point sampled from the latent space should give “meaningful” content once decoded.

Decoder
The Decoder reconstructs the original input from the representation of the latent space in order to generate new data. When given a point in the Latent space, the Decoder network works its magic, taking that abstract seed and skillfully mapping it back to the original input data space. The Decoder learns the art of reconstruction by minimizing the reconstruction error, striving to recreate the input data as faithfully as possible.

Variational Autoencoders (VAEs) have revolutionized unsupervised learning by combining the strengths of Autoencoders and Variational Inference. With their ability to learn latent representations and generate new data samples, VAEs have opened the possibility of tons of creative applications across domains. As the field continues to evolve, VAEs are likely to play a crucial role in advancing the frontiers of Machine Learning and Artificial Intelligence.

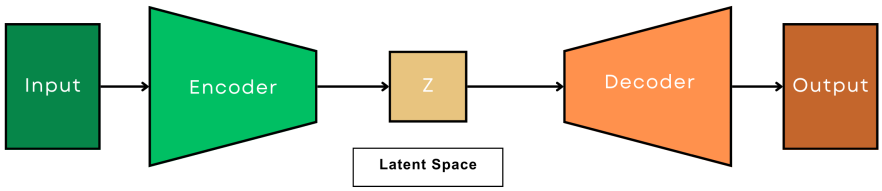

Figure 1.2: VAEs architecture. [15]

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) were first introduced by Ian Goodfellow in 2014 [26]. Generative Adversarial Networks have gained popularity in the field of image generation research. They are considered to generate the most realistic images among generative models [24]. GANs consist of two parts, a generator G and a discriminator D. The generator attempts to learn the distribution of real examples in order to generate new data, while the discriminator estimates the probability that a sample came from the training data set rather than the generated data set of G. [10]

The GANs architecture is based on the minimax two-player game, in which one player profits only when the other suffers an equal loss. The two players in GANs are the generator G and the discriminator D. The generative model G is pitted against an adversary: a discriminative model D that learns to determine whether a sample is from the model distribution (G) or the training data distribution. So, the generator's purpose is to trick the discriminator, while the discriminator's goal is to identify whether a sample is from a true data distribution (training data). [7] (see figure 1.3, page 20)

Therefore, the two parts (G,D) playing an adversarial game against each other. The generator, act as a counterfeiter and the discriminator you can think of as more of a detective. The generator creates fake data samples (images, audio, etc.) to deceive the discriminator. On the other hand, the discriminator seeks to discriminate between actual and fraudulent samples. [26]

The problem is formulated as a minimax game: D is trying to minimize the number of errors it makes while G is trying to maximize the number of errors D makes on generated samples. The value function V(G,D) (loss function) of the minimax game can be written as: [26]

$$\min_G \max_D V(D,G) = E_x[logD(x)] + E_z[log(1 - D(G(z)))]$$

A notation for the above function:

1. $D(x)$: is the discriminator's estimate of the probability that real data instance x is real.
2. E_x: is the expected value over all real data instances.
3. $G(z)$: is the generator's output when given noise z, as input.
4. $D(G(z))$: is the discriminator's estimate of the probability that a fake instance is real.
5. E_z: is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances $G(z)$).

The generator's training tries to minimize the term of the equation: $[log(1 - D(G(z)))]$.

The two parts of the GANs architecture train at the same time. During the adversarial training process, the generator and discriminator engage in a competition, wherein the generator strives to produce more realistic data and the discriminator aims to accurately classify whether the data is real or fake.

Key Components of GANs [26]

Generator
The training procedure for G is to maximize the probability of D making a mistake. The generator generates a batch of samples, and these, along with real examples from the domain, are provided to the discriminator and classified as real or fake. The generator part of a GAN learns to create fake data by taking feedback from the discriminator. Feedback from the discriminator helps the generator to improve its output over time. It learns to make the discriminator classify its output as real. So, we train the generator with the following procedure:

1. Sample random noise.
2. Produce generator output from sampled random noise.
3. Get discriminator "Real" or "Fake" classification for generator output.
4. Calculate loss from discriminator classification, $[log(1 - D(G(z)))]$.
5. Back-propagate through both the discriminator and generator to obtain gradients.
6. Use gradients to change only the generator weights.

Discriminator
The discriminator's goal is to identify whether a sample is from a true distribution or from the generated distribution. The discriminator's data comes from two sources:

1. Real data instances, such as real pictures of people, from the training data set. The discriminator uses these instances as positive examples during training.
2. Fake data instances created by the generator. The discriminator uses these instances as negative examples during training.

So, we train the discriminator with the following procedure:

1. The discriminator classifies both real data and fake data from the generator.
2. The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
3. The discriminator updates its weights through back-propagation from the discriminator loss through the discriminator network.

Diffusion models

Current research on diffusion models is mostly based on three predominant formulations: Denoising Diffusion Probabilistic Models (DDPMs), Score-Based Generative Models (SGMs), and Stochastic Differential Equations (Score SDEs). Key to all these approaches is to progressively perturb data with intensifying random noise (called the "diffusion" process), then successively remove noise to generate new data samples. [70] The essential idea, inspired by non-equilibrium statistical physics [59], is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. A denoising diffusion modeling is a two-step process [30]:

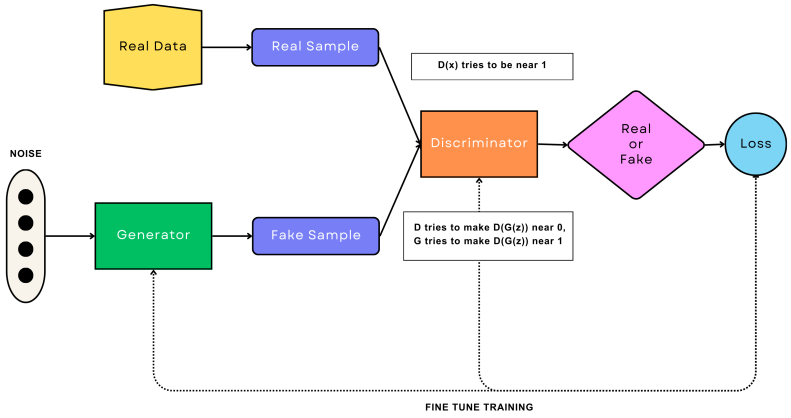

Figure 1.3: GANs architecture. [24]

Forward Diffusion Process

Given a data point sampled from a real data distribution $x_0 \sim q(x)$, the approximate posterior $q(x_{1:T}|x_0)$ called the forward process or diffusion process, is fixed to a Markov chain that gradually adds Gaussian noise to the sample in T steps producing a sequence of noisy samples $x_1,\ldots,x_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$:

$$q(x_{1:T}|x_0) := \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q(x_t|x_{t-1} := N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Reverse Diffusion Process

If we can reverse the above process and sample from $q(x_{t-1}|x_t)$, we will be able to recreate the true sample from a Gaussian noise input, $x \sim N(0, I)$. Unfortunately, we cannot easily estimate $q(x_{t-1}|x_t)$ because it needs to use the entire dataset and therefore we need to learn the joint distribution $p_\theta(x_{0:T})$ to approximate these conditional probabilities in order to run the reverse diffusion process:

$$p_\theta(x_{0:T}) := p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \qquad p_\theta(x_{t-1}|x_t) := N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Diffusion Models excel in sectors requiring high-resolution image generation, creative image stylization, video generation, and image inpainting, where the emphasis is on delivering aesthetically stunning and detailed outputs. [7]

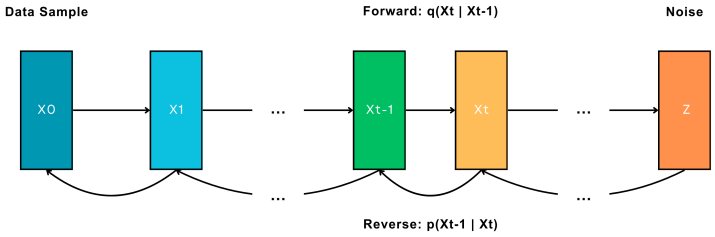

Figure 1.4: Diffusion Models architecture. [30]

Transformer-based models

Transformer was first explained in the paper Attention is All You Need by Google in 2017 [64]. Transformers, with their self-attention mechanism, efficiently capture long-term dependencies and have revolutionized natural language processing tasks. By creating coherent and contextually appropriate sequences, they excel at tasks like machine translation, text generation, and sentiment analysis. Transformers excel in sectors where comprehending global context and producing high-quality language sequences are essential.The architecture of the Transformer is based on encoder-decoder structure. 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. (see figure 1.5, page 23)

Key Components of Transformer [64]

Encoder

The encoder consists of a stack of N = 6 identical layers, where each layer is composed of two sub-layers. The first sub-layer implements a multi-head self-attention mechanism, and the second sub-layer is a fully connected feed-forward network. Furthermore, each of these two sub-layers has a residual connection around it followed by layer normalization. Each sub-layer is also succeeded by a normalization layer, layernorm(.), which normalizes the sum computed between the sub-layer input, X, and the output generated by the sub-layer itself, sub-layer(X): layernorm(X+ sublayer(X)) . The six layers of the Transformer encoder apply the same linear transformations to all the words in the input sequence, but each layer employs different weight W() and bias B() parameters to do so.

Decoder

The decoder shares several similarities with the encoder. The decoder also consists of a stack of N = 6 identical layers that are each composed of three sub-layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i. Hence, the prediction for a word at position i can only depend on the known outputs for the words that

come before it in the sequence.

There are also the following major parts on Transformer architecture:

1. **Attention Mechanism** [4]: A long time ago in the machine learning literature, the idea of incorporating a mechanism inspired by the human visual system into neural networks was introduced. This idea is named the attention mechanism. An attentional mechanism selectively focusing on parts of the source sentence during a task. An Encoder-Decoder kind of neural network architecture that allows the model to focus on specific sections of the input, where the most relevant information is concentrated, while executing a task.
2. **Self-Attention** [64]: Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.
3. **Multi-Head Attention** [64]: Multi-head Attention is a module for attention mechanisms which runs through an attention mechanism several times in parallel. The independent attention outputs are then concatenated and linearly transformed into the expected dimension.
4. **Feed-Forward Network** [25]: Deep feed-forward networks, also often called feed-forward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models. The goal of a feed-forward network is to approximate some function f^* . For example, for a classifier, $y = f^*(x)$ maps an input x to a category y. A feed-forward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters that result in the best function approximation. These models are called feed-forward because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself.
5. **Residual Connection** [28]: Residual connections are additional links that connect some layers in a neural network to other layers that are not directly adjacent.
6. **Layer Normalization** [3]: A simple normalization method to improve the training speed for neural network models. Normalizing inputs aims to create a set of features that are on the same scale.

Transformer models due to their faster training times and ability to process huge datasets with its more effective parallel computing dominate the area of generative AI. Stanford researchers called transformers “foundation models” in an August 2021 because they see them driving a paradigm shift in AI. [8]

1.1.3 Implementation phases of generative AI model

There are eight steps that you follow in order to build Generative AI models. These stages must be addressed in a systematic way to obtain the required results. Understanding these steps will help you create effective and reliable AI solutions. (see figure 1.6, page 24)

The initial phase consists of the problem definition. Before you start creating an AI-based solution, it’s worth spending some time on nailing down exactly what the problem is and the desired outcomes. Accurate problem definition facilitates targeted data collection. The subsequent phase focuses on the collection of data. The dataset should be diverse and comprehensive to capture the underlying patterns and characteristics that the generative model intends to learn. After data

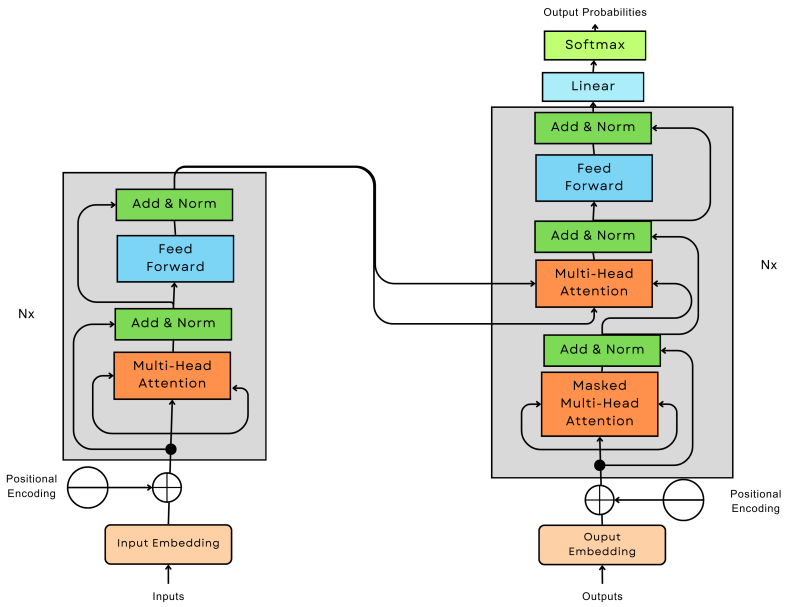

Figure 1.5: Transformer architecture: Encoder (left block) and the Decoder (right block). [64]

gathering, you have to select the appropriate D.L. architecture (VAEs, GANs, etc) which aligns with the problem requirements. Since you decide the architecture, the model training gets under way using the collected or available dataset. With this way the model learns the underlying patterns and statistical relationships within the data. Once the model training is completed, the subsequent stage involves evaluating its performance. You can use evaluation metrics that are tailored to the specific task or domain in order to assess the performance of the D.L. architecture. Probably, if things aren't going as well as expected after the evaluation, it's possible to fine-tune the model by adjusting the parameters or adding more data. Upon successful training and validation, the generative AI model is ready for deployment into the target environment in order to generate new content. Finally, you have to monitor and maintenance the model. Once you've rolled out your artificial intelligence model, it's important to keep close track of how it's doing in order to ensure that it still works efficiently. [6]

1.2 Classification of generative AI models

In this part, we introduce the categories in which we taxonomize current Generative AI models. Here we present a summary of the six different categories.

1.2.1 Text Generation

In the field of natural language processing, the ability to transform text into various textual outputs has been revolutionized by generative AI techniques. At the core of text-to-text generation techniques lies the process initiated by the user input or 'prompt'. When prompted, they can

Figure 1.6: Implementation phases of generative AI model. [6]

generate new content by rearranging and combining phrases. Such models can only generate content based on its training which is on vast amounts of text. The response can vary depending on the task: translation, answering questions, suggesting code corrections, or generating text in different fonts, the possibilities are vast.

Text models specially those centered around conversational chatbots have revolutionized AI since the launch of ChatGPT. Helped by natural language processing and large language models, these models have many very useful capabilities like summarization, writing assistance, code generation, language translation and sentiment analysis. They have been the main focus in Generative AI because of the capabilities of ChatGPT, an application which millions of users are already taking advantage of. The following table contains the related AI Text model generators. (see Table 1.1)

Input	Output	Prescribed Task	Model	Ref
text	text	Generate text in multiple languages	BLOOM	[69]
		Multimodal model which can accept image and text inputs and produce text outputs	GPT-4	[47]
		Taking a sequence of words as an input and predicts a next word to recursively generate text	LLaMA	[63]
		It has strong capabilities in multilingual tasks and source code generation	PaLM	[13]

Table 1.1: Text Generative AI Models

BLOOM's (BigScience Large Open-science Open-access Multilingual Language Model) [69] development was coordinated by BigScience, an open research collaboration whose goal was the public release of an LLM. BLOOM is an autoregressive Large Language Model (LLM), trained

to continue text from a prompt on vast amounts of text data using industrial-scale computational resources. BLOOM is a decoder-only Transformer language model that was trained on the ROOTS corpus, a dataset comprising hundreds of sources in 46 natural and 13 programming languages (59 in total). It's output is a coherent text that is hardly distinguishable from text written by humans. BLOOM can also be instructed to perform text tasks it hasn't been explicitly trained for, by casting them as text generation tasks.

BLOOM uses a Transformer architecture composed of an input embeddings layer, 70 Transformer blocks, and an output language-modeling layer, as shown in the figure below. Each Transformer block has a self-attention layer and a multi-layer perceptron layer, with input and post-attention layer norms. To predict the next token in a sentence using BLOOM, we simply need to pass the input tokens (in the form of embeddings) through each of 70 BLOOM blocks.

This is the culmination of a year of work involving over 1000 researchers from 70+ countries and 250+ institutions, leading to a final run of 117 days.

GPT-4 [47], was developed by OpenAI, is a large-scale, multimodal model which can accept image and text inputs and produce text outputs. It is used in a wide range of applications, such as dialogue systems, text summarization, and machine translation. GPT-4 is a Transformer-style model [64] pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF).

There are two kinds of model: an early version fine-tuned for instruction following ("GPT-4-early") and a version fine-tuned for increased helpfulness and harmlessness that reflects the further mitigations outlined in this system card ("GPT-4-launch").

The training process of the model is the following:
The GPT-4 model was trained to predict the next word in a document, and was trained using publicly available data (such as internet data) as well as data we've licensed. The data is a web-scale corpus of data including correct and incorrect solutions to math problems, weak and strong reasoning, self-contradictory and consistent statements, and representing a great variety of ideologies and ideas. So when prompted with a question, the base model can respond in a wide variety of ways that might be far from a user's intent. To align it with the user's intent within guardrails, we fine-tune the model's behavior using reinforcement learning with human feedback (RLHF).

GPT-4 is available on ChatGPT Plus and as an API for developers to build applications and services.

LLaMA (Large Language Model Meta AI) [63], was developed by Meta AI, is a collection of foundation language models ranging from 7B to 65B parameters. The model is available at several sizes 7B, 13B, 33B, and 65B parameters. The model is based on the Transformer's decoder-only architecture with the following modifications:

1. Pre-normalization: Model normalize the input of each transformer sub-layer, instead of normalizing the output.

2. SwiGLU activation function: Model replace the ReLU non-linearity by the SwiGLU activation function, to improve the performance.
3. Rotary Embeddings: Model remove the absolute positional embeddings, and instead, add rotary positional embeddings (RoPE), at each layer of the network.

LLaMA works by taking a sequence of words as an input and predicts a next word to recursively generate text. architecture. The training dataset of the models is a mixture of several sources, with the restriction of only using data that is publicly available. It was trained on Wikipedia Data, which covers 20 languages, most of the data the model was exposed to is from CommonCrawl, which contains English.

PaLM (Pathways Language Model) [13], was created by Google, is a text-to-text model that has strong capabilities in multilingual tasks and source code generation. PaLM uses a standard Transformer model architecture [64] in a decoder-only setup. There are 3 different models, the largest PaLM model has 540 billion dense parameters there is also trained 8 billion and 62 billion parameter models. The PaLM pretraining dataset consists of a high-quality corpus of 780 billion tokens that represent a wide range of natural language use cases. The dataset is a mixture of filtered webpages, 2 books, Wikipedia, news articles, source code, and social media conversations.

PaLM family of models was evaluated on a wide variety of tasks. Specifically, evaluation did on English Natural Language Processing tasks, tasks from BIG-bench [61] (Beyond the Imitation Game Benchmark) which includes over 150 tasks that cover a variety of language modeling tasks, reasoning tasks, code completion tasks, multilingual generation and question answering tasks, translation tasks, and bias and toxicity benchmarks. Some of the tasks that PaLM can do is the following:

1. goal step wikihow - The goal is to reason about goal-step relationship between events.
2. logical args – The goal is to predict the correct logical inference from a passage.
3. english proverbs – The goal is to guess which proverb best describes a text passage.
4. logical sequence – The goal is to order a set of “things” (months, actions, numbers, letters, etc.) into their logical ordering.
5. navigate – The goal is to follow a set of simple navigational instructions, and figure out where you would end up.
6. mathematical induction – The goal is to perform logical inference mathematical induction rules, even if they contradict real-world math.

The model was designed for research by Google.

1.2.2 Image Generation

This technology has proved very useful in creating images from text prompts as well as in image editing. In terms of art creation, it has pushed creative boundaries and has been revolutionary. AI Image generation uses artificial intelligence to generate images. One of the most well-known technique is the Text-to-image that has evolved significantly, enabling the generation of images from textual descriptions. Text-to-image AI models take inputs in the form of text prompts and produce an image matching the description using machine learning and deep neural networks. These models work by training on large datasets that contain both images and corresponding

textual descriptions. They learn to understand the relationship between specific words and phrases and the visual components they represent. When a user provides a new textual input, the model uses what it has learned to generate an image that it believes corresponds to the description. The following table contains the related AI Image model generators. (see Table 1.2)

Input	Output	Prescribed Task	Model	Ref
text	image	Generate images from text descriptions	DALL-E	[52]
		Generate images based on textual instructions	Imagen	[56]
		Generate images based on text and optional a simple sketch input	Make-A-Scene	[23]
image	image	Generate highly realistic and diverse synthetic images	StyleGAN	[32]

Table 1.2: Image Generative AI Models

DALL-E [52], was developed by OpenAI and first launched in January 2021 and generate digital images from natural languages descriptions called ”prompts”. It is a 12-billion parameter version of GPT-3 [9] trained on 250 million image-text pairs collected from the internet to generate images from text descriptions, using a dataset of text–image pairs. It has a diverse set of capabilities, including creating anthropomorphized versions of animals and objects, combining unrelated concepts in plausible ways, rendering text, and applying transformations to existing images.

DALL-E makes use of a transformer neural network to enable the model to create and understand connections between different concepts. It is a transformer language model. It receives both the text and the image as a single stream of data containing up to 1280 tokens (256 for the text and 1024 for the image) and is trained using maximum likelihood to generate all of the tokens, one after another. This training procedure allows DALL-E to not only generate an image from scratch, but also to regenerate any rectangular region of an existing image that extends to the bottom-right corner, in a way that is consistent with the text prompt.

DALL-E can generate imagery in multiple styles, including photorealistic imagery, paintings, and emojis. It can ”manipulate and rearrange” objects in its images and can correctly place design elements in novel compositions without explicit instruction.

In April 2022, OpenAI announced DALL-E 2 [51], a successor designed to generate more realistic images at higher resolutions that ”can combine concepts, attributes, and styles”. In October 2023, OpenAI announced DALL-E 3 [48], which is built on DALL-E 2 by improving caption fidelity and image quality.

Imagen [56] , was created by Google, is a text-to-image diffusion model that combines the power of transformer language models (LMs) with high-fidelity diffusion models. Imagen comprises a frozen T5-XXL [50] encoder to map input text into a sequence of embeddings and a 64×64 image diffusion model, followed by two super-resolution diffusion models for generating 256×256 and 1024×1024 images.

The process involves 2 essential components: Encoder and Diffusion Models. First, the caption is input into a text encoder. This encoder converts the textual caption to a numerical representation that encapsulates the semantic information within the text. Imagen utilizes a large frozen T5-XXL LLM encoder to encode the input text into embeddings. Next, we have several diffusion models to create the desired image. The process unfolds as follows:

1. An image-generation model creates an image by starting with noise, and slowly transforming it into an output image. To guide this process, the image-generation model receives the text encoding as an input, which has the effect of telling the model what is in the caption so it can create a corresponding image. The output is a 64x64 image that reflects visually the caption we input to the text encoder.
2. The 64x64 image is then passed into a super-resolution model, which grows the image to a higher resolution. This model also takes the text encoding as input, which helps the model decide how to behave as it ”fills in the gaps” of missing information that necessarily arise from quadrupling the size of our image. The result is a medium sized image, 256x256, of what we want.
3. The 64x64 image is then passed into a super-resolution model, which grows the image to a higher resolution. This model also takes the text encoding as input, which helps the model decide how to behave as it ”fills in the gaps” of missing information that necessarily arise from quadrupling the size of our image. The result is a medium sized image, 256x256, of what we want.

Each time the image is passed through a diffusion model, some noise is added. The denoising process, conditioned on the text prompt, is then applied to remove noise from the image, resulting in a more realistic and higher-quality image.

Make-A-Scene [23], was developed by Meta AI. The model generates a 2,048 x 2,048-pixel image given a text input and an optional scene layout. Like other generative AI models, Make-A-Scene learns the relationship between visuals and text by training on millions of example images. It is comprised of an autoregressive transformer, where in addition to the conventional use of text and image tokens, we introduce implicit conditioning over optionally controlled scene tokens, derived from segmentation maps.

Make-A-Scene’s architecture is comprised of an autoregressive transformer that uses text and image tokens as input. Additionally, the transformer uses conditionally controlled scene tokes using segmentation maps. During the inference phase, the segmentation tokens are generated from the input images or directly from the transformer. To effectively manage the loss, Make-A-Scene uses Vector-Quantized Variational Autoencoders (VQVAE) to encode and decode the image and scene tokens with explicit losses targeted at specific image regions correlated with human perception and attention.

Make-A-Scene aims to “realize AI’s potential to push creative expression forward,” according to Meta AI researchers.

StyleGAN [32], was created by NVIDIA, is a powerful technique in generating highly realistic and diverse synthetic images. Its architecture includes a generator network that produces synthetic images based on a learned mapping from a latent space to the image space. StyleGAN introduces

the concept of ”style” to GANs, which allows for better control over the generated images. The model generates images in a two-step process:

1. Mapping Network: The input to StyleGAN is a random noise vector $z \in Z$ space. This simply means that the given vector has arbitrary values from the normal distribution. This noise vector is first passed through a mapping network, which learns to convert it into a latent space representation (W space). This latent space representation is a vector that encodes the style or visual properties of the generated image. The mapping network is an 8-layer MLP (Multilayer Perceptron) that takes in and outputs 512-dimensional vectors.
   (a) Latent Space: We can think of it as a space where each image is represented by a vector of N dimensions. The goal is to get unique information from each dimension. In this way, the latent space would be disentangled, and the generator would be able to perform any wanted edits on the image.
2. Synthesis Network: The latent space representation is then used as input to the synthesis network, which generates the image. The synthesis network employs a series of convolutional layers, with different resolutions and styles. The style information is incorporated into the network's different layers, allowing for control over various image attributes, such as color, texture, and structure.

The StyleGAN is an extension of the progressive growing GAN that is an approach for training generator models capable of synthesizing very large high-quality images via the incremental expansion of both discriminator and generator models from small to large images during the training process.

1.2.3 Audio Generation

Text-to-speech (TTS) and text-to-audio techniques both involve converting text into human-like speech and generating music with various vocal styles, respectively. TTS is commonly used in applications such as voice assistants, voice navigation systems, and audio-books. The following table contains the related AI Audio model generators. (see Table 1.3)

Input	Output	Prescribed Task	Model	Ref
text	audio	Generate music that can be conditioned on an artist, genres and lyrics	Jukebox	[18]
		Generate high-fidelity music pieces from text descriptions	MusicLM	[1]
		Generate highly realistic, human-like speech in a variety of languages and accent	VALL-E	[66]
		Produces high-quality audio clips	Voicebox	[36]

Table 1.3: Audio Generative AI Models

Jukebox [18] is a model that generates raw audio music imitating many different styles and artists. Provided with genre, artist, raw audio and lyrics as input, Jukebox outputs a new music sample

produced from scratch. It was realized by the OpenAI team and trained on 1.2 million songs (600k of which in English), paired with the lyrics and metadata from LyricWiki and music pieces by various musicians, composers, and bands.

Jukebox is made of 2 parts, (1) Compressing music to discrete codes and (2) Generating codes using transformers. It tackles the long context of raw audio which take as input using a hierarchical VQ-VAE [53] architecture to compress audio into a discrete codes and modelling those discrete codes using autoregressive Transformers. The autoregressive models are trained using a simplified variant of Sparse Transformers. [12]

First, a hierarchical VQ-VAE architecture is used to compress audio into a discrete codes. There are three levels in VQ-VAE scheme, which compress the 44kHz raw audio by 8x, 32x, and 128x respectively. This down sampling loses much of the audio detail, and sounds noticeably noisy as we go further down the levels. However, it retains essential information about the pitch, timbre, and volume of the audio. Each VQ-VAE level independently encodes the input. The bottom level (8x) encoding produces the highest quality reconstruction, while the top level(128x) encoding retains only the essential musical information.

Next, it follows the training of the prior models, which they follow the architecture of Sparse Transformers, whose goal is to learn the distribution of music codes encoded by VQ-VAE and to generate music in this compressed discrete space. Like the VQ-VAE, we have three levels of priors: a top-level prior that generates the most compressed codes, and two up-sampling priors that generate less compressed codes conditioned on above. The top-level prior models the long-range structure of music, and samples decoded from this level have lower audio quality but capture high-level semantics like singing and melodies. The middle and bottom up-sampling priors add local musical structures like timbre, significantly improving the audio quality.

MusicLM [1], by Google, can generate high-fidelity music pieces from text descriptions. Using hierarchical sequence-to-sequence modeling task, MusicLM has the capability to generate music at 24 kHz that remains consistent over several minutes from text descriptions. The model has the ability to be conditioned on melodies, such as a melody that is then synthesized according to the text prompt. This is the Melody conditioning. As a result, MusicLM can generate music based on both a text description and a melody, which is provided in the form of humming, singing, whistling, or playing an instrument.

Model consists of Soundstream [71], w2v-BERT [14], MuLAN [31]. All three of these parts are pre-trained independently and they provide the discrete audio and text representations for the sequence-to-sequence modeling. Following the explanation of each part of the model:

1. Soundstream: Generate acoustic tokens to enable high-fidelity synthesis. The result is one second of audio is represented by 600 tokens.
2. w2v-BERT: Generate semantic tokens to facilitate long-term coherent generation. The result is 25 semantic tokens for every second of audio.
3. MuLAN: For representing the conditioning, model rely on the MuLan where is used to construct a music-text joint embedding. It was trained on pairs of music clips and their corresponding textual annotations. This process yields 12 MuLan audio tokens for an audio sequence.

During training, the model learns to convert the token mapping produced by MuLan to semantic tokens (w2w-BERT). Then the acoustic token (Soundstream) is conditioned on both the MuLan audio tokens and the semantic tokens (SoundStream). During inference the process is to provide a textual description to MuLAn which transforms it into conditional signaling, this in turn is transformed into an audio token by w2w-BERT and then transformed into waveforms by the SoundStream.

MusicLM can generate long audio sequences where the textual description changes over time. This approach is called ”story mode”.

VALL-E [66], was developed by Microsoft, a language modelling approach to text-to-speech synthesis. That is, it can speak the given text in that voice by listening to your voice for only 3 seconds. VALL-E will be able to learn to extract relevant speaker vocal identity, emotion etc. from a just a small audio prompt, apply that to a text prompt to generate personalized speech. VALL-E is a text-to-speech model that resembles language models in its operational mode, such that it predicts the next discrete audio token for a given prompt, which consists of phonemicized text and audio input. VALL-E generates the discrete audio codec codes based on phoneme and acoustic code prompts, corresponding to the target content and the speaker’s voice.

Microsoft researchers describe the core element of the VALL-E as neural codec language modelling. The input of the VALL-E model is a phonemicized text and a 3-second audio sample and the output are the corresponding sound waveform. This allows the generation of a speech utterance of the input text, which is conditioned on the given audio prompt, this means that the model generate speech from a voice unseen in the training data. First, there is a phoneme conversion of the text, which is standard procedure and doesn’t require any learning mechanism. Furthermore, the 3-second acoustic prompt, which the output speech is conditioned on, is fed into an audio codec encoder. VALL-E uses a pre-trained audio encoder for this, Encodec, which was developed by Facebook Research [19]. Encodec takes as input a waveform of speech and outputs a compressed discrete representation of it . Neural net Encodec has 3 parts:

1. Encoder: transforms raw data into higher dimensional and lower frame rate.
2. Quantizer: compresses to target size, equiv. to mp3.
3. Decoder: turns compressed signal back to waveform, most similar to the original.

Once the model receives these two inputs (a phonemicized text and a 3-second audio sample), it can act as an autoregressive language model and output the next discrete audio representation. Because the audio representations come from a fixed vocabulary which was learned by Encodec, we can think of this simply as predicting the next word in a sentence out of a fixed vocabulary of words (a fixed vocabulary of sound representations, in our case). After these sound representations are predicted they are transformed back into the original waveform representation using the Decoder part of the Encodec model. VALL-E could keep the acoustic environment and speaker’s emotion in synthesis and provide diverse outputs in different sampling-based decoding processes.

Voicebox [36], was developed by Meta AI researchers, produces high-quality audio clips. The model can synthesize speech across six languages, as well as perform noise removal, content editing, style conversion, and diverse sample generation. It is based on a method called Flow Matching [38], which is Meta’s latest advancement on non-autoregressive generative models that can learn highly non-deterministic mapping between text and speech. Non-deterministic mapping

is useful because it enables Voicebox to learn from varied speech data without those variations having to be carefully labeled. Voicebox is a non-autoregressive flow-matching model trained to infill speech given audio context and text. There is an English-only Voicebox version which was trained on 60K hours of data and a multilingual version on 50K hours of data covering six languages (English, French, German, Spanish, Polish, and Portuguese).

Voicebox is trained to predict a speech segment when given the surrounding speech and the transcript of the segment. Having learned to infill speech from context, the model can then apply this across speech generation tasks, including generating portions in the middle of an audio recording without having to re-create the entire input. This versatility enables Voicebox to perform well across a variety of tasks. It is capable of:

1. Creating speech that sounds like a given sample. Using an input audio sample just two seconds in length, Voicebox can match the sample's audio style and use it for text-to-speech generation.
2. Creating speech in different languages. Given a sample of speech and a passage of text in English, French, German, Spanish, Polish, or Portuguese, Voicebox can produce a reading of the text in that language.
3. Cleaning up and editing noisy or mistaken speech. Voicebox's in-context learning makes it good at generating speech to seamlessly edit segments within audio recordings. It can resynthesize the portion of speech corrupted by short-duration noise, or replace misspoken words without having to rerecord the entire speech.
4. Creating new samples that sound like real people talking. Having learned from diverse in-the-wild data, Voicebox can generate speech that is more representative of how people talk in the real world and across the six languages listed above.

1.2.4 Video Generation

Video Generative AI helps producers with storytelling. Although still a developing field because of the complexity that video generation poses, listed use cases such as digital human videos, human motion capture and video dubbing are revolutionary uses which can quickly lead to technological change. Text-to-video models, as the name suggests, use natural language prompts as input to generate a video. These models use advanced machine learning or deep learning techniques or a recurrent neural network to understand the context and semantics of the input text and then generate a corresponding video sequence. We can also have image-to-video or video-to-video generative models. The following table contains the related AI Video model generators. (see Table 1.4)

Gen-1 [20], by Runway, is a ground-breaking AI model that transforms existing videos into new ones using text prompts or reference images. Gen-1 present a structure and content-guided video diffusion model that edits videos based on visual or textual descriptions of the desired output. The trained model can be further customized to generate more accurate videos of a specific subject by fine-tuning on a small set of images.

The model separates the video into two things. It will be helpful to think of a video in terms of its content and structure. By structure, we refer to characteristics describing its geometry and

Input	Output	Prescribed Task	Model	Ref
text	video	Transforms existing videos into new ones using text prompts or reference images	Gen-1	[20]
		Generate text-guided videos	ImagenVideo	[29]
			Make-A-Video	[58]
text+image	video	Generating video from text prompt and input image	Phenaki	[65]

Table 1.4: Video Generative AI Models

dynamics, e.g. shapes and locations of subjects as well as their temporal changes. By content, we refer to features describing the appearance and semantics of the video, such as the colors and styles of objects and the lighting of the scene. The goal of the model is then to edit the content of a video while retaining its structure. The generative process of the model is the following:

1. First, similar to image synthesis models, model is trained such that the content of inferred videos, e.g. their appearance or style, match user-provided images or text prompts.
2. Second, inspired by the diffusion process, an information obscuring process is applied to the structure representation to enable selecting of how strongly the model adheres to the given structure.
3. Finally, the inference process is adjusted via a custom guidance method, inspired by classifier-free guidance, to enable control over temporal consistency in generated clips.

The model is trained jointly on images and videos. It was trained on a large-scale dataset of uncaptioned videos and paired text-image data.

Imagen Video [29], introduced by Google. It's a text-conditional video generation system based on diffusion models. Given a text prompt, Imagen Video generates high-definition videos with high frame fidelity, strong temporal consistency, and deep language understanding using a base video generation model and a sequence of interleaved spatial and temporal video super-resolution models. The model is capable of generating videos with artistic styles learned from image information, such as videos in the style of van Gogh paintings or watercolour paintings. Also, it possesses an understanding of 3D structure, as it is capable of generating videos of objects rotating while roughly preserving structure. Furthermore, Imagen Video is also reliably capable of generating text in a wide variety of animation styles, some of which would be difficult to animate using traditional tools.

Imagen Video, it consists of 7 sub-models which perform text-conditional video generation, spatial super-resolution, and temporal super-resolution. We have 1 frozen text encoder, 1 base video diffusion model, 3 SSR (spatial super-resolution), and 3 TSR (temporal super-resolution) models. The SSR models increase spatial resolution for all input frames, whereas the TSR models increase temporal resolution by filling in intermediate frames between input frames. The components and techniques that constitute Imagen Video are described below:

1. The Data gets into the Input Text Prompt and then gets into the T5-XXL (frozen text encoder) model to perform the contextual encoding of the text prompt. This allows the system to have a deeper language understanding.

2. It's then sent to the Base video model that generates the base structure of the video with a low resolution and a low frame rate.
3. Then there is a series of TSR and SSR. TSR is filling the gap between the frame to get a higher frame rate while SSR is used to improve the resolution guided by the Input text prompt.

By extending the text-to-image diffusion models of Imagen [56] to the time domain, and training jointly on video and images, Imagen Video is a model capable of generating high fidelity videos with good temporal consistency while maintaining the strong features of the original image system.

Make-A-Video [58] by Meta AI, is a text-to-video generation AI model. Make-A-Video is trained using publicly available text-image pairs and video-only data. The Meta researchers was inspired from the following things in order to create the model:

1. Its difficult to build from scratch text-to-video synthesis models due to the limited dataset for training the models.
2. They already exist text-to-image models that can generate images.
3. Moreover, unsupervised learning enables networks to learn from orders of magnitude more data.

Unlike some other text-to-video (T2V) models, Make-a-Video does not require a dataset of text-video pairs. Instead, it is based on existing text-image pair models, which generate single-frame images from a text description. Generated images are expanded in both spatial and temporal dimension using an additional series of neural network layers. The Meta researchers instead opted to use a pre-trained encoder based on CLIP, which they call the prior, as the base of their model. This converts the input text into an image embedding, followed by a decoder that converts that image embedding into a series of 16 frames of 64x64 pixel images. The team then used unsupervised learning on video data without text labels to learn a model to up sample the generated images to a higher frame rate and pixel resolution. The architecture of the model is consisted of the following three primary components:

1. the text-to-image model trained on text-image pairs,
2. the Spatio-temporal layer in which there are two layers, the convolution and attention layer,
3. and the network of interpolation and the spatio-temporal layer.

The T2I model trained on text-image pairs. This network produces high-resolution images from text. The Spatio-temporal layer is responsible to generate videos through convolutional layers and attention layers. The interpolation network can increase the number of frames of the generated video either by frame interpolation for a smoother generated video or by pre or post frame extrapolation for extending the video length.

Phenaki [65], was created by Google, is a model capable of realistic video synthesis, given a sequence of textual prompts. What makes it special is that Phenaki creates a coherent video - a story - from successive text inputs. That is the model can synthesize realistic videos from textual prompt sequences. It is auto-regressive in time. This allows for generating long videos, while the prompt changes over time. Time variable prompts can be thought of as a story, a narration of the entire video where each prompt corresponds to a scene from the video. This allows for creating dynamically changing scenes.

It is a video generation model that utilizes a bidirectional transformer architecture. By leveraging textual descriptions as input, it has the capability to generate video sequences. The model excels in generating videos that correspond to different time-varying text prompts, allowing for dynamic and diverse output. Phenaki is designed with two main components:

1. An encoder-decoder model that compresses videos to discrete embeddings, or tokens, with a tokenizer that can work with variable-length videos thanks to its use of causal attention in time.
2. A transformer model that translates text embeddings to video tokens: we use a bi-directional masked transformer conditioned on pre-computed text tokens to generate video tokens from text, which are subsequently de-tokenized to create the actual video.

The multi-modal model was trained primarily with text-image pairs. In addition, the researchers trained Phenaki with 1.4-second short video-text pairs at eight frames per second.

1.2.5 Code Generation

In the world of software development, the task of manually writing or replicating code patterns can be a time-consuming process. However, there is an innovative solution known as text-to-code, which offers significant benefits. Text-to-code enables us to generate entire source code for specific business problems, resolve issues within existing code by providing suggestions for corrections. It is essential to address the range of tasks that generative AI models can accomplish in the field of code generation. These tasks include generating valid programming code using natural language descriptions, which can be achieved through various models. These models excel in converting natural language descriptions into executable code. Text-to-code AI models use machine learning to generate snippets of code or entire functions. These models are trained on vast amounts of public code and are designed to aid human developers. They take natural language inputs – plain English – and can turn it into code. The following table contains the related AI Code model generators. (see Table 1.5)

Input	Output	Prescribed Task	Model	Ref
text	code	Generate valid programming code using natural language query	CodeBERT	[21]
		Generate valid programming code using natural language description in a multi-step process.	CODEGEN	[46]
		Generate valid programming code using natural language descriptions	CodeT5	[68]
		Generate valid programming code using natural language prompts and is most capable in Python	Codex	[11]

Table 1.5: Code Generative AI Models

CodeBERT [21], is a bimodal pre-trained model for natural language (NL) and programming language (PL), developed by Microsoft Research. This model has been trained on NL-PL pairs

in 6 programming languages (Python, Java, JavaScript, PHP, Ruby, Go). CodeBert is built with transformer-based neural architecture [64] and utilizes both natural languages and source codes as its input.

The model captures the semantic connection between natural language and programming language, and produces general-purpose representations that can broadly support NL-PL understanding tasks (e.g. natural language code search) and generation tasks (e.g. code documentation generation). There are four major use cases of the model, which are the following:

1. Code-to-Code Translation: Converting code from one PL to another, such as translating Python code to Java.
2. Code-to-Text: Generating human-readable summaries of code snippets to aid in code comprehension and documentation.
3. Text-to-Code: Generating relevant code based on natural language query.
4. Text-to-Text: Translating code domain text to different languages.

CodeBERT is trained with both bimodal data, which refers to natural language-code pairs, and unimodal data, which stands for codes without paired natural language texts and natural language without paired codes. Each bimodal datapoint is an individual function with paired documentation, and each unimodal code is a function without paired documentation.

CodeGen [46] is a family of autoregressive-transformer language models that specializes in program synthesis based on input and expected output or even natural language descriptions. The goal of program synthesis is to automate the coding process and generate a computer program that satisfies the user's specified intent. The paper [46] proposes a multi-step paradigm for program synthesis, where a single program is factorized into multiple prompts specifying sub-problems. The program synthesis is decomposed into multiple steps and the system synthesizes a sub-program in each step. To solve a problem, a model needs to synthesize functionally correct sub-programs following the description at the current step and considering descriptions and synthesized sub-programs at previous steps. So, a user communicates with the synthesis system by progressively providing specifications in natural language while receiving responses from the system in the form of synthesized sub-programs, such that the user together with the system complete the program in multiple steps.

The family of CODEGEN models is trained sequentially on three datasets: THEPILE, BIGQUERY, and BIGPYTHON.

1. The natural language dataset THEPILE is English text, in the majority. The models trained on this dataset are called as natural language CODEGEN models (CODEGEN-NL).
2. The multi-lingual dataset BIGQUERY consists of code in multiple programming languages. The models trained on this dataset are called as multi-lingual CODEGEN models (CODEGEN-MULTI).
3. The mono-lingual dataset BIGPYTHON contains a large amount of data in the programming language, Python. The models trained on this dataset are called as mono-lingual CODEGEN models (CODEGEN-MONO).

The CODEGEN models are in the form of autoregressive transformers with next-token prediction language modelling as the learning objective trained on a natural language corpus and program-

ming language data. Also, the models are trained in various sizes of parameters.

CodeT5 [68], is a unified pre-trained encoder–decoder transformer model based on the T5 architecture [50], capable of tasks such as code understanding, code generation, and converting source code between programming languages. It aims to derive generic representations for programming language (PL) and natural language (NL) via pre-training on unlabeled source code. The model leverages the code semantics conveyed from the developer-assigned identifiers.

The training datasets are source code including user-written comments from open source Github repositories and publicly available. At the pre-training stage, model would receive two kinds of input either PL-only or NL-PL as inputs depending on whether the code snippet has accompanying NL descriptions or not. For the NL-PL bimodal inputs, we concatenate them into a sequence with a delimiter token between the number of NL word tokens and PL code tokens. The NL word sequence will be empty for PL-only unimodal inputs.

There are two types of models based on the parameters' number, CodeT5-small (60M) and CodeT5-base (220M). CodeT5 can do various types of downstream tasks, such as:

1. Code Summarization: summarize a function-level code snippet into english description.
2. Code Generation: the model generates code snippet in various PL languages, based on NL description.
3. Code-to-Code Generation: there are two subtasks that model can do, code translation and code refinement.
   (a) In code translation, model convert code from one PL to another, i.e. convert C# to Java code and vice versa.
   (b) In code refinement, detect which parts of code are buggy and fix them via generating a bug-free code sequence.

Codex [11] is a powerful AI model developed by OpenAI, is a general-purpose programming model, meaning that it can be applied to essentially any programming task. Codex, a GPT language model fine-tuned on publicly available code from GitHub, and is most capable in Python. Developed by the OpenAI team, Codex uses GPT3 architecture [9] and tokenizer, and pre-trains on a large corpus of Github code. This large language model was one of the first successful Code LLM to generate code from doc-string or natural language prompts with high accuracy and was a state-of-art model in 2021. Codex has the potential to be useful in a range of ways. For example, it could help onboard users to new codebases, reduce context switching for experienced coders, enable non-programmers to write specifications and have Codex draft implementations, and aid in education and exploration. Codex will generate code that is as similar as possible to its training distribution.

1.2.6 3D Generation

Generative AI models have revolutionized the field of image synthesis, allowing for the generation of visually compelling 2D images based on textual descriptions. However, the potential of these models extends beyond 2D images, as they can also be applied to the domain of 3D content generation. In this section, we will discuss various text-to-3D, image-to-3D and 3D-to-3D techniques.

These technologies allow for easier 3D designs just with having a text prompt, an image or a video. They have varied applications such as game creation, the metaverse or urban planning for which 3D designs are fundamental. 3D model generation can be achieved through many types of inputs (text, image, images and 2D models) through generative AI. AI 3D generation uses artificial intelligence to generate 3D objects. The following table contains the related AI 3D model generators. (see Table 1.6)

Input	Output	Prescribed Task	Model	Ref
text	3D	Generate 3D objects based on textual descriptions	Dreamfusion	[49]
		Generate 3D images using textual descriptions	Magic3D	[37]
		Generate 3D point clouds from text prompts	Point-E	[45]
image	3D	Generate high quality 3D model from any viewpoint given a single image	ImageDream	[67]

Table 1.6: 3D Generative AI Models

DreamFusion [49] was developed by Google which leverage the combined power of two things, Imagen [56] and NeRF(Neural Radiance Field) [41]. More specifically, Imagen is a pre-trained text-to-image diffusion model which is used to optimize a 3D scene and NeRF (Neural Radiance Field) is a model used for rendering 3D scenes by generating a neural radiance field from one or more images of an object. This approach basically tries to generate not a 3d image but create an image from all possible angles that a camera might cover. Basically, we are trying to create a 3D model by generating images of all possible angles a camera could cover, looking around the object, and guessing the pixels' colours, densities, light reflections.

The generative process of the Dreamfusion is the following:

1. A random NeRF field is initialized and trained for each caption.
2. The NeRF computes the density of the shade scene and the light directions. Shading is important as it reveals important details about the geometry of a 3D object.
3. Uses Imagen to predict a denoised image and construct a better image.
4. Backpropages an update to the NeRF weights and optimize.
5. Repat until it converges.

Generative models such as ours may have the potential to displace creative workers via automation. That said, these tools may also enable growth and improve accessibility for the creative industry.

Magic-3D [37] was developed by Nvidia and it is a pre-trained text-to-image diffusion model. It can create quality 3D mesh models from text descriptions of objects. The model focus on text-to-3D synthesis, aiming to generate a 3D renderable representation of a scene based on a text prompt. It makes use of a coarse-to-fine technique to learn the 3D representation of the target material by combining low- and high-resolution diffusion priors. Magic3D can create high-quality 3D mesh models in 40 minutes.

Magic3D utilizes a two-stage process:

1. In the first stage, it uses a low-resolution diffusion prior to producing a coarse model, which it then accelerates using a hash grid and sparse acceleration structure. The base diffusion model described in eDiff-I [5]. This diffusion prior is used to compute gradients of the scene model.
2. In the second stage, the model employs a textured mesh model that is initialized from the coarse neural representation to enable optimization using a high-resolution latent diffusion model in conjunction with an effective differentiable renderer. It uses the publicly available Stable Diffusion model [54].

Nvidia Magic3D relies at its core on a generative image model that uses text to generate images from different angles, which in turn serve as input for 3D generation. Nvidia hopes as the tech evolves it can be used for asset creation in video games and VR development.

Point-E [45], was created by OpenAI, a model for generating 3D point clouds from complex text prompts. The model is comprised of two models – text-to-image and image-to-3D. In other words, when you type in a text query, such as "a corgi wearing a Santa hat", the text-to-image model will find generate a related image. Then, the image-to-3D model will produce a 3D object based on the sampled image. Point-E combines the benefits of both categories by pairing a text-to-image model with an image-to-3D model. The text-to-image model leverages a large corpus of (text, image) pairs, allowing it to follow diverse and complex prompts, while the image-to-3D model is trained on a smaller dataset of (image, 3D) pairs.

The generation process can break down into three steps. First, model generate a synthetic view conditioned on a text caption. Next, it produces a coarse point cloud (1,024 points) conditioned on the synthetic view. And finally, it produces a fine point cloud (4,096 points) conditioned on the low-resolution point cloud and the synthetic view. In practice, it assume that the image contains the relevant information from the text, and do not explicitly condition the point clouds on the text. To generate text-conditional synthetic views, a text prompt is fed into a GLIDE [44] model to produce a synthetic rendered view. To generate low-resolution point clouds, it uses a conditional, permutation invariant diffusion model [72]. To up-sample these low-resolution point clouds, it uses a similar (but smaller) diffusion model which is additionally conditioned on the low-resolution point cloud.

Text-to-3D generation has significant potential to democratize 3D content creation for a wide range of applications such as virtual reality, gaming, and industrial design. The model has the ability to support the creation of point clouds that can then be used to fabricate products in the real world, for example through 3D printing.

ImageDream [67] was developed by ByteDance Researchers. It is an advanced image-prompt 3D generation model, transform 2D images into 3D models using a process called multi-view diffusion. This involves generating multiple 2D images from different perspectives, which are then synthesized into a cohesive 3D representation. For example, suppose you input a picture of a bulldog wearing a black pirate hat. In that case, ImageDream generates multiple views of the object and then uses those multiple views to create a 3D model.

ImageDream-approach involves considering a canonical camera coordination across different object instances and designing a multi-level image-prompt controller that can be seamlessly integrated into the existing architecture. Specifically, the canonical camera coordination mandates that the rendered image, under default camera settings, represents the object's centered

front-view. This significantly simplifies the task of mapping variations in the input image to 3D. The multilevel controller offers hierarchical control, guiding the diffusion model from the image input to each architectural block, thereby streamlining the path of information transfer.

ImageDream excels in generating objects with correct geometry from a given image. As technology continues to evolve, tools like ImageDream will become even more integral to our digital lives, blending creativity with technology to open up new realms of possibility. From gaming and VR to product design and beyond, the possibilities are endless.

The Generative models can be classified based on deep learning architecture. All the models have the same input type, which is natural language description. (see figure 1.7, page 40)

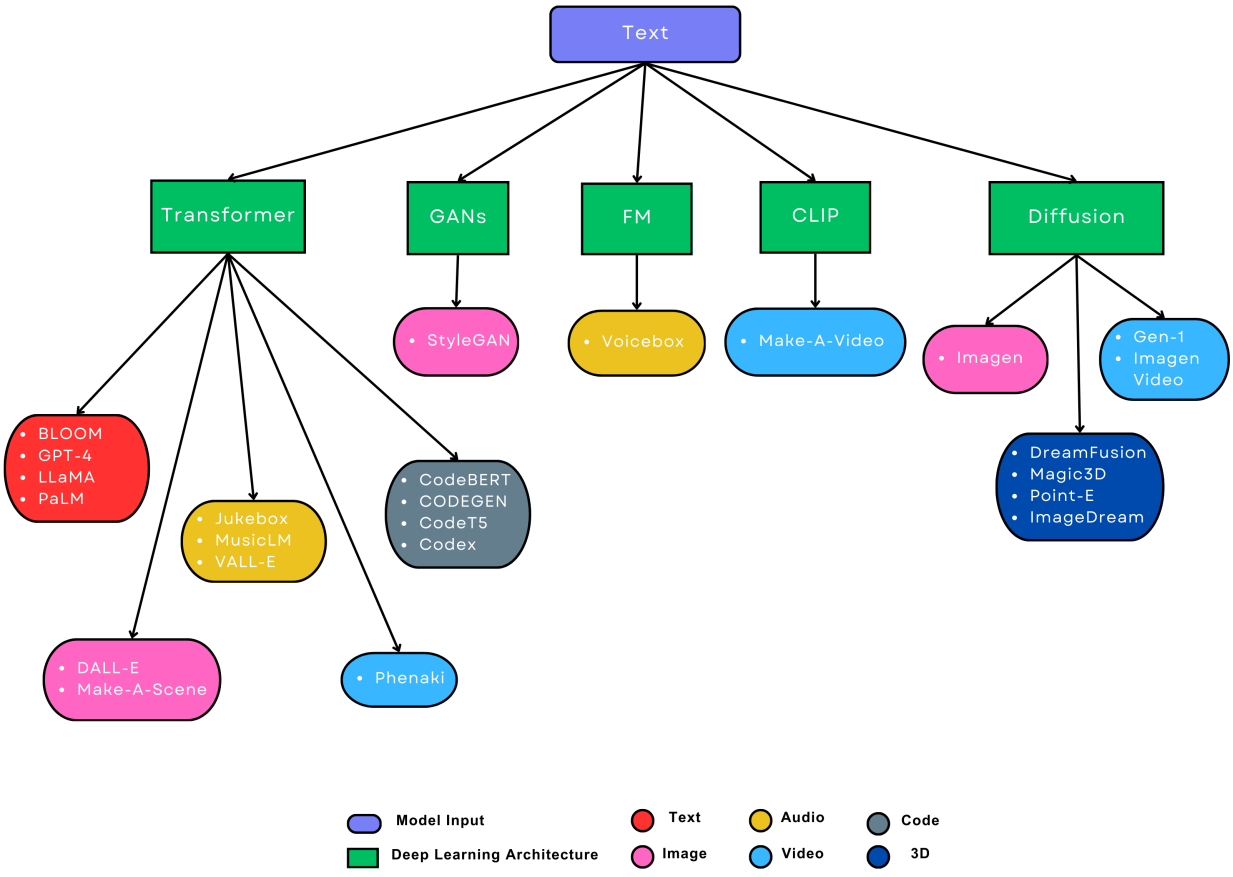

Figure 1.7: Tree of Models

2. EVALUATION

We perform a human evaluation comparing the most common open-source (Stable Diffusion, Crayon(DALLE mini)), and commercial (DALL-E 2) models. These models have 3 common hyperparameters: prompt, height, width. For the evaluation of these three models, we provide a benchmark of seven task-types with twelve prompts of each task type.

2.1 Models

As we said we will perform a human evaluation for the following three models.

DALL-E 2

DALL-E 2 is created by OpenAI and is a successor of DALL-E. It can create more realistic images than DALL-E at higher resolutions and can combine concepts, attributes, and styles. Model is trained on approximately 650 million image-text pairs scraped from the Internet. The images of DALLE-2 were created through OpenAI API. Python code was written and OpenAI API was used in order to create images based on the caption. **TODO REFERENCE**

Crayon

Craiyon (previously DALL·E mini) is an AI model that can generate images from any prompt you give. As the model developers wrote in the project report about DALL·E mini, "OpenAI had the first impressive model for generating images with DALL·E. DALL·E mini is an attempt at reproducing those results with an open-source model." [16]

The model is trained by looking at millions of images from the internet with their associated captions. The model was trained on unfiltered data from the Internet, limited to pictures with English descriptions. Text and images from communities and cultures using other languages were not utilized. This affects all output of the model, with white and Western culture asserted as a default, and the model's ability to generate content using non-English prompts is observably lower quality than prompts in English. The model developers used 3 datasets for the model: [17]

1. Conceptual Captions Dataset, which contains 3 million image and caption pairs.
2. Conceptual 12M, which contains 12 million image and caption pairs.
3. The OpenAI subset of YFCC100M, which contains about 15 million images and that we further sub-sampled to 2 million images due to limitations in storage space. They used both title and description as caption and removed html tags, new lines and extra spaces.

The model is 27 times smaller than the original DALL-E with about 0.4 billion parameters.

Stable Diffusion

Stable Diffusion is a text-to-image latent diffusion model created by the researchers and engineers from CompVis, Stability AI and LAION.This is a model that can be used to generate and modify images based on text prompts. Model is based on a particular type of diffusion model called Latent Diffusion, proposed in High-Resolution Image Synthesis with Latent Diffusion Models [54]

We use the The Stable-Diffusion-v1-5 checkpoint, that was initialized with the weights of the Stable-Diffusion-v1-2 checkpoint and subsequently fine-tuned on 595k steps at resolution 512x512 on "laion-aesthetics v2 5+" . [54], [55]

In order to run the stable diffusion model, we installed the AutomaticGui1111, a web interface for Stable Diffusion, implemented using Gradio library. [2]

2.2 Benchmark

We provide a multi-task benchmark for evaluating the text-to-image models. Our benchmark contains a suite of tasks over multiple applications that capture a model's ability to handle different features of a text prompt. Our benchmark contains seven task types:

1. colouring task: The prompt is used to examine the ability of the image generation system to generate images that have colouring concepts.
2. counting task: The prompt is used to examine the ability of the image generation system to generate images that have counting concepts.
3. conflicting task: The prompt is used to examine the ability of the image generation system to generate images that have conflicting concepts.
4. text task: The prompt is used to examine the ability of the image generation system to generate images that have texting concepts.
5. positional task: The prompt is used to examine the ability of image generation systems to generate images with accurate positional information.
6. shapes task: The prompt is used to examine the ability of image generation systems to generate images with accurate shaping information.
7. faces task: The prompt is used to examine the ability of image generation systems to generate images with accurate facing information.

Each task type contains 12 prompts and each task type consists of three difficulty levels (easy, medium, and hard). Our evaluation protocol consists of human ratings. For each prompt, the rater is shown 3 sets of images with one from DALL-E 2, second from Stable Diffusion, and third from Craiyon. The human rater will be asked two questions:

1. 1. Which set of images better represents the text caption: [Text Caption]? Question subjectively evaluates image-text alignment.
2. 2. Which set of images is of higher quality? Question subjectively evaluates image fidelity.

For each question, the rater is asked to select from three choices:

1. I prefer set A

2. I prefer set B
3. I prefer set C

The scores from different raters will be aggregated and then score it in a percentage value which will be presented in the form of a graph.

Below is the benchmark per task type.

Level	**Prompt**
Easy (1 coloured item)	A red coloured car.
	A black coloured car.
	A pink coloured car.
	A navy blue coloured car.
Medium (2 coloured item)	A red car and a white sheep.
	A blue bird and a brown bear.
	A green apple and a black backpack.
	A green cup and a blue cell phone.
Hard (3+ coloured item)	A red pencil in a green cup on a blue table.
	An office with five desks and seven colourful chairs.
	An orange bird scaring a blue scarecrow with a red pirate hat.
	Two pink football balls and three green basketball balls on a bench.

Table 2.1: Colouring Tasks

Level	**Prompt**
Easy (1-3 number of objects)	One tennis ball on the court.
	Three people crossing the street.
	Two monkeys eating banana.
	Bench in a park with two backpacks on it.
Medium (4-10 number of objects)	Nine children doing a circle dance around a Christmas tree.
	An office desk with six laptops on it.
	Person holding four toy pyramids.
	Five photograph prints are hanging to dry in a dark room.
Hard (10+ number of objects)	Person carrying a stack of twelve books.
	Two people juggling ten balls together.
	Birthday cake with exactly twenty candles on it.
	Office room with fifteen chairs.

Table 2.2: Counting Tasks

Level	Prompt
Easy	A zebra without strips.
	A penguin in a city.
	A panda inside a cup of tea.
	An ant under the sea.
Medium	A man walking on the ceiling.
	A giraffe inside a car.
	Rainbow coloured Ferrari.
	A giraffe kissing a monkey.
Hard	A polar bear on the desert.
	A motorcycle inside an oven.
	A fish eating a pelican.
	A horse riding an astronaut in the forest.

Table 2.3: Conflicting Tasks

Level	Prompt
Easy	A sign that says 'Hello'.
	A sign that says 'World'.
	A sign that says 'Hello World'.
	A sign that says 'World Hello'.
Medium	A sign that says 'Speed limit 45'.
	A sign that says 'Do not pass!'.
	A sign that says 'No pedestrians'.
	A sign that says 'No overtaking'.
Hard	A sign that says 'No Goods materials'.
	A sign that says 'No hazardous materials'.
	A sign that says 'You must not turn Right'.
	A sign that says 'Maximum speed limit 80 km'.

Table 2.4: Text Tasks

Level	Prompt
Easy	A train on top of a surfboard.
	A wine glass on top of a dog.
	A bicycle on top of a boat.
	An umbrella under a spoon.
Medium	A car on the left of a bus.
	A black apple on the right of a green backpack.
	A carrot on the left of a broccoli.
	A pizza on the right of a suitcase.
Hard	A cat on the right of a tennis racket.
	A stop sign on the right of a refrigerator.
	A sheep to the right of a wine glass.
	A zebra to the right of a fire hydrant.

Table 2.5: Positional Tasks

Level	Prompt
Easy (simple shapes)	Circle shape.
	Triangle shape.
	Star shape.
	Hexagon shape.
Medium (entity in the form of shape)	Eraser in the shape of a star.
	Lamp shaped like a diamond.
	TV shaped like an octagon.
	Diamond shaped bush.
Hard (multiple entities with shapes)	A triangular tree with heart shaped leaves.
	Five-pointed star medal and a cubic chair.
	An octagon shaped cookie on a heart shaped plate.
	Square shaped water bottle next to a semicircular orange.

Table 2.6: Shape Tasks

Level	Prompt
Easy (faces with 1-2 features)	Face of a man with a goatee.
	Face of an angry teacher.
	Face of a kid with a tiger make-up.
	Face of an old lady with blue hair and a wide smile.
Medium (1-2 faces with 1-3 features)	A bald man doing a handstand.
	Face of a woman with brown hair looking over her shoulder.
	Sad face of a blonde girl holding a popped balloon.
	Face of a weightlifting white hair Olympic gold medalist lifting 120kg with a referee next to it encouraging him.
Hard (2+ faces with 2+ features)	The face of a soldier with camouflaged face painting obscured by leaves.
	A face of a man with a beard with water splashing onto his face.
	A dark messy hair boy rolling his tongue with blue light shinning on his face.
	A young kid with dark eye bags standing in front of her grandma with wrinkles looking at the sky.

Table 2.7: Face Tasks

3. CONCLUSION

ABBREVIATIONS - ACRONYMS

AI	Artificial Intelligence
D	Discriminator
DNN	Deep Neural Network
DL	Deep Learning
DDPMs	Denoising Diffusion Probabilistic Models
FM	Flow Matching
GPT	Generative Pre-trained Transformer
GAI	Generative Artificial Intelligence
GANs	Generative Adversarial Networks
G	Generator
LLM	Large Language Model
ML	Machine Learning
NL	Natural Language
NN	Neural Network
PL	Programming Language
SSR	Spatial Super-Resolution
SGMs	Score-Based Generative Models
SDEs	Stochastic Differential Equations
TSR	Temporal Super-Resolution
T2V	Text-to-Video
T2I	Text-to-Image
TTS	Text to Speech
VAEs	Variational Auto-Encoders
2D	2 Dimensions
3D	3 Dimensions

REFERENCES

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023.

[2] AUTOMATIC1111. stable-diffusion-webui, 2023.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[5] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers, 2023.

[6] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. The power of generative ai: A review of requirements, models, input-output formats, evaluation metrics, and challenges. *Future Internet*, 15(8), 2023.

[7] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. The power of generative ai: A review of requirements, models, input-output formats, evaluation metrics, and challenges. *Future Internet*, 15(8), 2023.

[8] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro

Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[10] Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun. A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt, 2023.

[11] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[12] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

[13] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[14] Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training, 2021.

[15] Lucas Cinelli, Matheus Marins, Eduardo da Silva, and Sergio Netto. *Variational Methods for Machine Learning with Applications to Deep Networks*. 01 2021.

[16] Boris Dayma, Suraj Patil, Pedro Cuenca, Khalid Saifullah, Tanishq Abraham, Phuc Le Khac, Luke Melas, and Ritobrata Ghosh. Dall·e mini, 7 2021.

[17] Boris Dayma, Suraj Patil, Pedro Cuenca, Khalid Saifullah, Tanishq Abraham, Luke Phúc Lê, and Ritobrata Ghosh. Weights and biases, 2022.

[18] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

[19] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression, 2022.

[20] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models, 2023.

[21] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. Codebert: A pre-trained model for programming and natural languages, 2020.

[22] Stefan Feuerriegel, Jochen Hartmann, Christian Janiesch, and Patrick Zschech. Generative ai. *Business, Information Systems Engineering*, September 2023.

[23] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors, 2022.

[24] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks, 2017.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[26] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[27] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchán. A survey of generative ai applications, 2023.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[29] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022.

[30] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[31] Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis. Mulan: A joint embedding of music audio and natural language, 2022.

[32] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[33] Yashar Kiarashinejad, Sajjad Abdollahramezani, and Ali Adibi. Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures. *npj Computational Materials*, 6(1), February 2020.

[34] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[35] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[36] Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, and Wei-Ning Hsu. Voicebox: Text-guided multilingual universal speech generation at scale, 2023.

[37] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation, 2023.

[38] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.

[39] Yue Liu, Zhengwei Yang, Zhenyao Yu, Zitu Liu, Dahui Liu, Hailong Lin, Mingqing Li, Shuchang Ma, Maxim Avdeev, and Siqi Shi. Generative artificial intelligence and its applications in materials science: Current situation and future perspectives. *Journal of Materiomics*, 9(4):798–816, 2023.

[40] Zhihan Lv. Generative artificial intelligence in the metaverse era. *Cognitive Robotics*, 3:208–217, 2023.

[41] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[42] Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[43] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[44] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models, 2022.

[45] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts, 2022.

[46] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis, 2023.

[47] OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis,

Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.

[48] OpenAI. Dall-e 3 system card, 2023.

[49] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022.

[50] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena,

Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[51] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[52] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[53] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019.

[54] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[55] Robin Rombach and Patrick Esser. Huffing face, stable-diffusion-v1-5 model card, 2023.

[56] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[57] Haroon Sheikh, Corien Prins, and Erik Schrijvers. *Mission AI: The New System Technology*. Springer International Publishing, 2023.

[58] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data, 2022.

[59] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[60] C. O. S. Sorzano, J. Vargas, and A. Pascual Montano. A survey of dimensionality reduction techniques, 2014.

[61] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo,

Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman,

Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023.

[62] The European Commission. *A definition of AI: Main capabilities and scientific disciplines*. The European Commission, 2019.

[63] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[65] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description, 2022.

[66] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, Lei He, Sheng Zhao, and Furu Wei. Neural codec language models are zero-shot text to speech synthesizers, 2023.

[67] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation, 2023.

[68] Yue Wang, Weishi Wang, Shafiq Joty, and Steven C. H. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation, 2021.

[69] BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del

Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie

Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. Bloom: A 176b-parameter open-access multilingual language model, 2023.

[70] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2024.

[71] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec, 2021.

[72] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion, 2021.