**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCES**
**DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS**

**THEORETICAL INFORMATICS**

**MSc THESIS**

# Evaluation of three text-to-image Gen AI models

**Evgenios N. Mazarakis**

**Supervisors: Theoharis Theoharis**, Professor NKUA

**ATHENS**

**JUNE 2024**

**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**
**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΘΕΩΡΗΤΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# Evaluation of three text-to-image Gen AI models

**Ευγένιος Ν. Μαζαράκης**

**Επιβλέποντες: Θεοχάρης Θεοχάρης**, Καθηγητής ΕΚΠΑ

**ΑΘΗΜΑ**

**ΙΟΥΝΙΟΣ 2024**

**MSc THESIS**

Evaluation of three text-to-image Gen AI models

**Evgenios N. Mazarakis**
**S.N.:** M1458

**SUPERVISORS: Theoharis Theoharis**, Professor NKUA

**EXAMINATION COMMITTEE: Theoharis Theoharis**, Professor NKUA
**Roussou Maria**, Associate Professor NKUA
**Ntoulas Alexandros**, Assistant Professor NKUA

**Examination Date: 28 June 2024**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

Evaluation of three text-to-image Gen AI models

**Ευγένιος Ν. Μαζαράκης**
**Α.Μ.:** M1458

**ΕΠΙΒΛΕΠΟΝΤΕΣ: Θεοχάρης Θεοχάρης**, Καθηγητής ΕΚΠΑ

**ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: Θεοχάρης Θεοχάρης**, Καθηγητής ΕΚΠΑ
**Ρούσσου Μαρία**, Αναπληρώτρια Καθηγήτρια ΕΚΠΑ
**Ντούλας Αλέξανδρος**, Επίκουρος Καθηγητής ΕΚΠΑ

**Ημερομηνία Εξέτασης: 28 Ιουνίου 2024**

## 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 3D-scenes.

The investigation begins with an 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 and evaluation of three models (DALL-E 2, Stable Diffusion, Pix-Art-α) in the graphics domain based on some parameters. Executing generative models through Python code or running them through an interface empowers users to harness advanced AI techniques. Furthermore, the evaluation of these models involves both qualitative and quantitative processes. 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, Deep Learning

## 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 τρισδιάστατες σκηνές.

Η διπλωματική εργασία ξεκινά με μια εξερεύνηση των Variational Autoencoders, 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 και αξιολόγηση τριών μοντέλων (DALL-E 2, Stable Diffusion, Pix-Art-α) στον τομέα των γραφικών με βάση ορισμένες παραμέτρους. Η εκτέλεση των μοντέλων μέσω κώδικα Python ή εκτελώντας μέσω μιας διεπαφής δίνει τη δυνατότητα στους χρήστες να αξιοποιήσουν προηγμένες τεχνικές AI. Επιπλέον η αξιολόγηση των μοντέλων γίνεται με ποιοτική και ποσοτική μέθοδο. 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, Βαθιά Μάθηση

*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 Ellie 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. I would like to thank my supervisor Mr. Theoharis Theoharis for his trust and his valuable time which along with his guidance were a great help to the completion of the study. Much gratitude to my cousin Nicky for inspecting the thesis and correcting spelling and grammatical errors. Last but not least, I would like to thank the 17 people, my colleagues from Performance Technologies, and also my friends who helped me with the completion of the questionnaires.

## CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# PREFACE

The aim of the thesis is to focus on Generative AI algorithms. The core focus of the thesis lies in executing and evaluating the capabilities of three text-to-image generative AI models.

First, we have an introduction for the area of the Generative AI. In the second chapter, we provide all the necessary definitions commonly used for the AI, Generative AI and Generative AI model, that are the necessary background we need so as to understand what is mentioned in this thesis. Besides, we talk about some fundamental deep learning architectures that are used on generative AI models. In addition, we describe the implementation phases of a Generative AI models, and then we offer a classification of some of the advanced GAI algorithms. Finally, the categorization of the models has been done based on the generated content, the output they create.

In the third chapter, we elucidate the three under examination models, namely, the prevalent open-source frameworks (Stable Diffusion and Pix-Art-α), as well as the commercial counterpart, DALL-E 2. These models share three common hyperparameters: prompt, height and width, thus guiding our experimental setup. Afterwards, we conducted model executions based on the above parameters. To conduct qualitative and quantitative assessment of these models, we present a benchmark comprising six task-types, each consisting of twelve prompts.

The fourth chapter of the thesis contains the outcomes of the qualitative and quantitative assessment stemming from the implementation of Generative AI models for graphics. Additionally, notable findings related to the previously mentioned models in the graphics field are pointed out.

The final chapter of the thesis is the conclusion drawn from the evaluation of three text-to-image models.

# ΠΡΟΛΟΓΟΣ

Στόχος της παρούσας διπλωματικής εργασίας είναι να εστιάσει στους αλγορίθμους Generative AI. Η ουσία της εργασίας εστιάζει στην εκτέλεση και αξιολόγηση των δυνατοτήτων τριών μοντέλων τεχνητής νοημοσύνης που μετατρέπουν κείμενο σε εικόνα.

Αρχικά έχουμε μια εισαγωγή για την περιοχή της γενεσιουργού τεχνητής νοημοσύνης. Στο δεύτερο κεφάλαιο, 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.

Στο τρίτο κεφάλαιο, διευκρινίζουμε τα τρία υπό εξέταση μοντέλα, δηλαδή, τα ανοιχτού κώδικα (Stable Diffusion and Pix-Art-α), μαζί με το αντίστοιχο εμπορικό, DALL-E 2. Αυτά τα μοντέλα μοιράζονται τρεις κοινές υπερπαραμέτρους: κείμενο (prompt), ύψος (height) και πλάτος (width), καθοδηγώντας έτσι την πειραματική μας διαδικασία. Στη συνέχεια, εκτελέσαμε τα μοντέλα με βάση αυτές τις παραμέτρους. Για τη διεξαγωγή ποιοτικής και ποσοτικής αξιολόγησης αυτών των μοντέλων, παρουσιάζουμε ένα σημείο αναφοράς που περιλαμβάνει έξι τύπους εργασιών, καθένας από τους οποίους αποτελείται από δώδεκα περιπτώσεις κειμένου (prompt).

Στο τέταρτο κεφάλαιο, παρουσιάζονται τα αποτελέσματα της ποιοτικής και ποσοτικής αξιολόγησης ύστερα από την εκτέλεση των μοντέλων για τα γραφικά. Επιπλέον, αναφέρονται αξιοσημείωτες παρατηρήσεις που σχετίζονται με αυτά τα τρία μοντέλα στον τομέα των γραφικών.

Το τελευταίο κεφάλαιο της εργασίας είναι τα συμπεράσματα που προέκυψαν από την αξιολόγηση των τριών μοντέλων.

# 1. INTRODUCTION

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 modelling 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. Furthermore, 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. [27]

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 [31]. In 2017, Google researchers develop the concept of transformers in the research paper "Attention is All You Need" [77]. The Transformer model, is a novel architecture that revolutionized the field of natural language processing (NLP) and became the basis for the large language models (LLMs) as 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. [27]

The organization of this thesis includes the following chapters: The second chapter provides comprehensive definitions within the domain of Generative AI, discusses important deep learning techniques, and outlines the implementation phases of a Generative AI model. It also presents a taxonomy of AI models categorized by their generated content and discusses evaluation methodologies. The third chapter introduces the three text-to-image models assessed in this study, detailing their execution based on specified parameters such as prompt, width, and height. Additionally, this chapter presents the benchmark utilized for the qualitative and quantitative assessment of the models and describes the two techniques used to evaluate the models' performance. The fourth chapter discusses the assessment results, highlighting findings derived from the model executions. In the fifth and final chapter, we present the conclusions, summarizing the key findings and discussing their implications. The structure of the master's thesis is illustrated in the following figure:


Figure 1.1: Thesis organization

# 2. BACKGROUND AND RELATED WORK

## 2.1 Background on Generative AI

In this section, we address several fundamental concepts related to artificial intelligence, including definitions of generative AI, models, and large language models (LLMs). Subsequently, we discuss four widely recognized deep learning techniques. Finally, we outline the implementation phases of a generative AI model.

### 2.1.1 General Definitions

**Definition 2.1.1** (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. [75]

**Definition 2.1.2** (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. [32]

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

**Definition 2.1.3** (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**. [49]

**Definition 2.1.4** (Model)**.** A formal description of a system, process or equation used to simplify a complex subject. [71]

**Definition 2.1.5** (Discrim. Model)**.** Discriminative models determine a conditional probability to accomplish classification and decision-making given data. [47]

**Definition 2.1.6** (Gen. Model)**.** Generative models directly predict a distribution and generate new data. [47]

**Definition 2.1.7** (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. [30]

**Definition 2.1.8** (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. [30]

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 2.1.9** (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. [30]

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 2.1.10** (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. [50]

**Definition 2.1.11** (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. [50]

**Definition 2.1.12** (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. [50]

**Definition 2.1.13** (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. [50]

**Definition 2.1.14** (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. [21]

**Definition 2.1.15** (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). [27]

**Definition 2.1.16** (Unimodal)**.** Unimodal (Single-modal) models take instructions from the same input type as their output. [27]

**Definition 2.1.17** (Multimodal)**.** Multimodal models can take input from different sources and generate output in various forms. [27]

### 2.1.2 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 architectures are Generative Adversarial Networks (GANs), Variational Auto-encoders (VAEs), Diffusion and Transformer-based models. A brief introduction to these well known techniques is what this section is about.

#### 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. [74]

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)). [39]

**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 to uncover hidden patterns, structures, or relationships within the data. 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. [39]

**Introducing Variational Auto-Encoders**
The Variational Autoencoder (VAE) [41] 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 2.1) [46]

**Key Components of VAEs** [41], [40]

***Encoder***
In the world of VAEs, the Encoder acts as an interpreter who not only comprehends the substance of the input data, it also captures the complex 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.

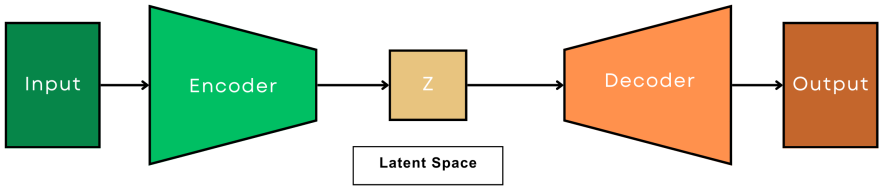


Figure 2.1: VAEs architecture. [15]

### Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) were first introduced by Ian Goodfellow in 2014 [31]. 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 [29]. 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. [11]

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). [8] (see figure 2.2)

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. [31]

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: [31]

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

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** [31]

***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.

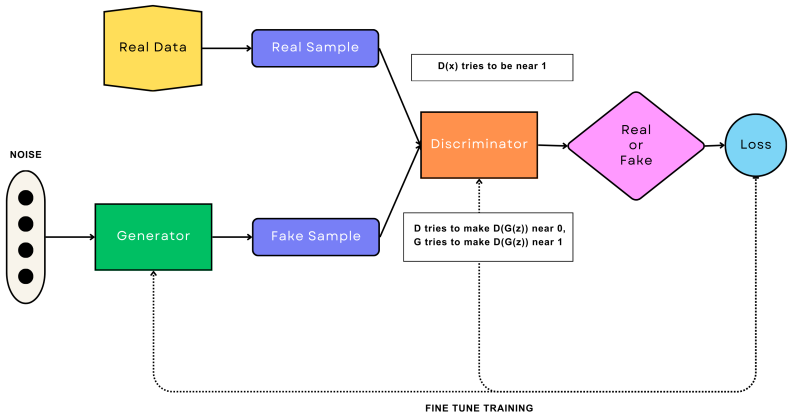


Figure 2.2: GANs architecture. [29]

### 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. [83] The essential idea, inspired by non-equilibrium statistical physics [73], 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 [36] (see figure 2.3) :

**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 $x_1$,…,$x_T$. 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. [8]

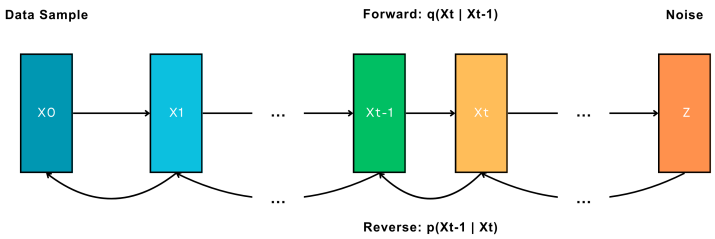


Figure 2.3: Diffusion Models architecture. [36]

**Transformer-based models**

Transformer was first explained in the paper Attention is All You Need by Google in 2017 [77]. 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 2.4)

**Key Components of Transformer** [77]

***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 the 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** [5]: 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** [77]: 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** [77]: 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** [30]: 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** [33]: Residual connections are additional links that connect some layers in a neural network to other layers that are not directly adjacent.
6. **Layer Normalization** [4]: 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. [21]

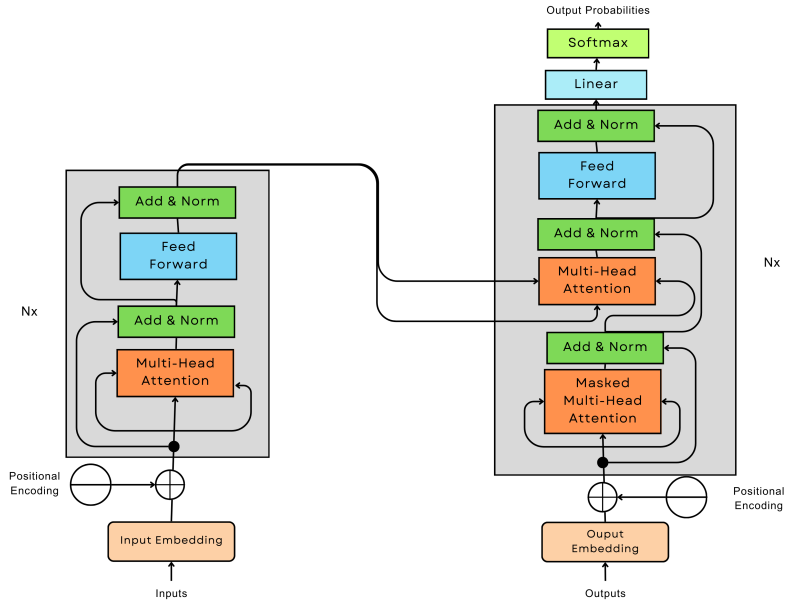


Figure 2.4: Transformer architecture: Encoder (left block) and Decoder (right block). [77]

### 2.1.3 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 2.5)

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 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. [7]


Figure 2.5: Implementation phases of generative AI model. [7]

## 2.2 Related Work on Generative AI models

In this section, we categorize generative AI models into six categories based on their output types. Additionally, we discuss four models from each category. Furthermore, we provide an overview of the evaluation methods used to assess the text-to-image models.

### 2.2.1 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 generate new content, by rearranging and combining phrases. The response can vary depending

on the task: translation, answering questions, suggesting code corrections, or generating text in different fonts.

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 lots of 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 the ChatGPT, an application used by millions of users. The following table contains the related AI Text model generators. (see Table 2.1)

| Input | Output | Prescribed Task | Model | Ref |
|---|---|---|---|---|
| text | text | Generate text in multiple languages | BLOOM | [82] |
| | | Multimodal model which can accept image and text inputs and produce text outputs | GPT-4 | [56] |
| | | Taking a sequence of words as an input and predicts a next word to recursively generate text | LLaMA | [76] |
| | | It has strong capabilities in multilingual tasks and source code generation | PaLM | [24] |

Table 2.1: Text Generative AI Models

**BLOOM's** (BigScience Large Open-science Open-access Multilingual Language Model) [82] 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). Its 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** [56], 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 [77] 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) [76], 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. 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) [24], 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 [77] 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 [25] (Beyond the Imitation Game Benchmark) which includes over 150 tasks that cover a variety of language modeling tasks, reason-

ing 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.

### 2.2.2 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 2.2)

| Input | Output | Prescribed Task | Model | Ref |
|---|---|---|---|---|
| text | image | Generate images from text descriptions | DALL-E | [65] |
| | | Generate images based on textual instructions | Imagen | [23] |
| | | Generate images based on text and optional a simple sketch input | Make-A-Scene | [28] |
| image | image | Generate highly realistic and diverse synthetic images | StyleGAN | [38] |

Table 2.2: Image Generative AI Models

**DALL-E** [65], 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 [20] 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 [64], 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 [55], which is built on DALL-E 2 by improving caption fidelity and image quality.

**Imagen** [23] , 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 [63] 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** [28], was developed by Meta AI. The model generates a 2048 x 2048-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** [38], 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.

### 2.2.3 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 2.3)

| Input | Output | Prescribed Task | Model | Ref |
|---|---|---|---|---|
| text | audio | Generate music that can be conditioned on an artist, genres and lyrics | Jukebox | [17] |
| | | Generate high-fidelity music pieces from text descriptions | MusicLM | [2] |
| | | Generate highly realistic, human-like speech in a variety of languages and accent | VALL-E | [79] |
| | | Produces high-quality audio clips | Voicebox | [42] |

Table 2.3: Audio Generative AI Models

**Jukebox** [17] 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 [66] 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. [13]

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** [2], 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 [84], w2v-BERT [14], MuLAN [37]. 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** [79], 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 [18]. 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, equivalent 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** [42], 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 [44], 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.

### 2.2.4 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 2.4)

| Input | Output | Prescribed Task | Model | Ref |
|---|---|---|---|---|
| text | video | Transforms existing videos into new ones using text prompts or reference images | Gen-1 | [19] |
| | | Generate text-guided videos | ImagenVideo | [35] |
| | | | Make-A-Video | [72] |
| text+image | video | Generating video from text prompt and input image | Phenaki | [78] |

Table 2.4: Video Generative AI Models

**Gen-1** [19], 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 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** [35], 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 [23] 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** [72] 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. 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** [78], 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.

### 2.2.5 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 2.5)

| Input | Output | Prescribed Task | Model | Ref |
|---|---|---|---|---|
| text | code | Generate valid programming code using natural language query | CodeBERT | [26] |
| | | Generate valid programming code using natural language description in a multi-step process. | CODEGEN | [53] |
| | | Generate valid programming code using natural language descriptions | CodeT5 | [81] |
| | | Generate valid programming code using natural language prompts and is most capable in Python | Codex | [22] |

Table 2.5: Code Generative AI Models

**CodeBERT** [26], 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 [77] 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** [53] 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 [53] 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 programming language data. Also, the models are trained in various sizes of parameters.

**CodeT5** [81], is a unified pre-trained encoder–decoder transformer model based on the T5 architecture [63], 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** [22] 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 [20] 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.

### 2.2.6 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 2.6)

| Input | Output | Prescribed Task | Model | Ref |
|---|---|---|---|---|
| text | 3D | Generate 3D objects based on textual descriptions | Dreamfusion | [61] |
| | | Generate 3D images using textual descriptions | Magic3D | [43] |
| | | Generate 3D point clouds from text prompts | Point-E | [52] |
| image | 3D | Generate high quality 3D model from any viewpoint given a single image | ImageDream | [80] |

Table 2.6: 3D Generative AI Models

**DreamFusion** [61] was developed by Google which leverage the combined power of two things, Imagen [23] and NeRF(Neural Radiance Field) [48]. 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** [43] 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 [6]. 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 [67].

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** [52], 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 (1024 points) conditioned on the synthetic view. And finally, it produces a fine point cloud (4096 points) conditioned on the low-resolution point cloud and the synthetic view. In practice, it assumes that the image contains the relevant information from the text, and does not explicitly condition the point clouds on the text. To generate text-conditional synthetic views, a text prompt is fed into a GLIDE [51] model to produce a synthetic rendered view. To generate low-resolution point clouds, it uses a conditional, permutation invariant diffusion model [85]. 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.

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** [80] was developed by ByteDance Researchers. It is an advanced image-prompt 3D generation model, which transforms 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 2.6)

### 2.2.7 Evaluation Methodologies

There are two approaches to assess generative models: quantitative and qualitative methods. Initially, quantitative methods involve calculating numerical scores based on specific criteria. Text-to-image models are commonly evaluated using the Inception Score (IS) and the Fréchet Inception Distance (FID). Another metric is the CLIP Score, which is used to define the image-text alignment of the generative model. Quantitative methods can provide objective and standardized measures of generative model performance, but they also have some limitations, such

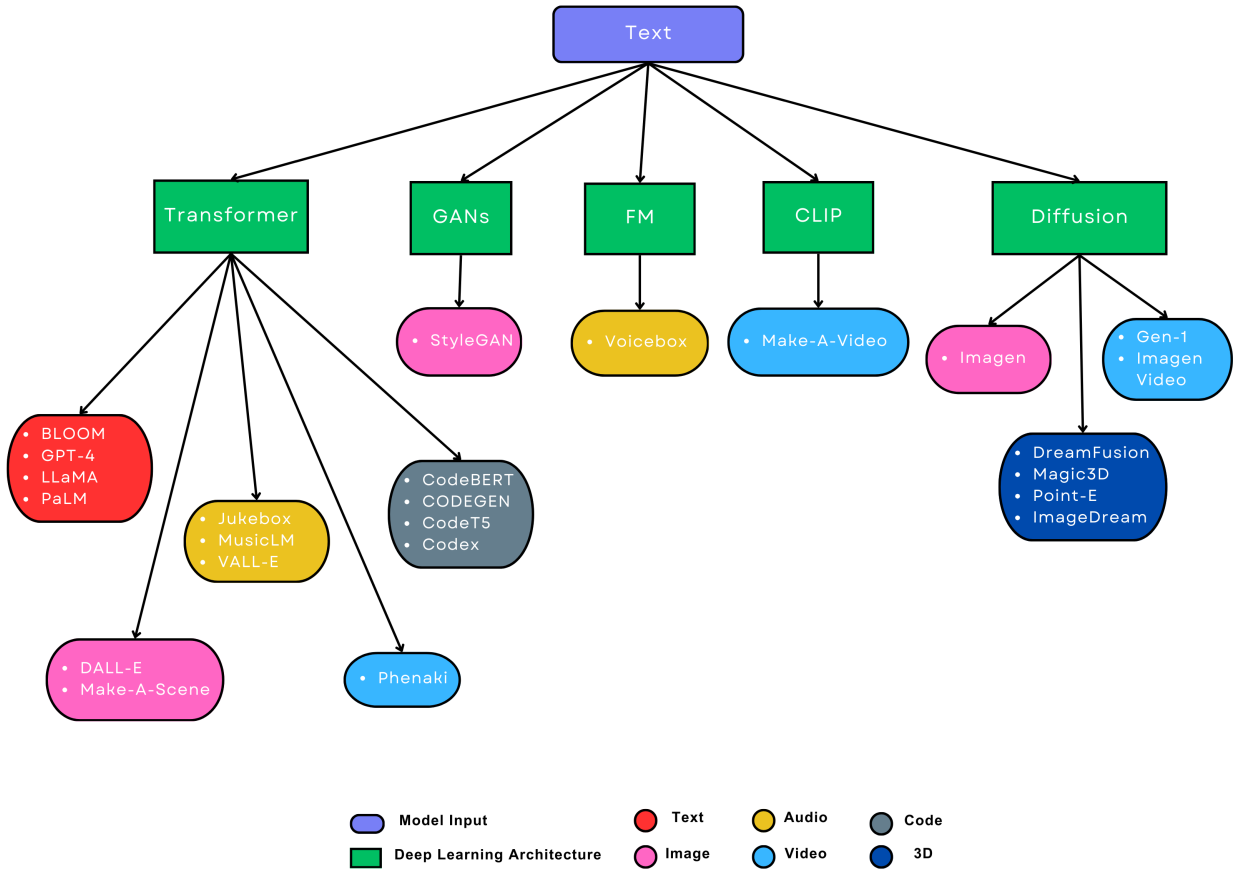


Figure 2.6: Classification of models

as requiring a reference dataset, being sensitive to model architecture, or being difficult to interpret.

The Inception Score, first defined in a 2016 technical paper [70] and is based on Google's "Inception" image classification network. The Inception Score is a metric for automatically evaluating the quality of image generative models. The IS uses an Inception v3 Network, a deep convolutional architecture, pre-trained on ImageNet, a dataset consisting of 1.2 million RGB images from 1000 classes. Given an image x, the task of the network is to output a class label y in the form of a vector of probabilities $p(y|x) \in [0,1]^{1000}$ , indicating the probability the network assigns to each of the class labels [9]. A probability distribution is simply a numbered list of what the image classification network "thinks" the image might be – each with a fractional score that adds up to 1.0.

Frechet Inception Distance (FID) is one of the most popular metrics for measuring the feature distance between real and generated images. FID was first introduced to evaluate the performance of Generative Adversarial Networks (GANs) [34]. It is used to measure the discrepancy between two image sets: I and I'. Typically, one set consists of real images, and the other set is generated by the image generation model being evaluated. The metric employs the squared Fréchet distance between two probability distributions.

The CLIP Score [62] is a well-known metric for evaluating the similarity between a generated image and its related prompt. CLIP, short for Contrastive Language-Image Pretraining, is a deep learning model created by OpenAI in 2021. CLIP is designed to identify which of the N × N possible (image, text) pairings within a batch are true matches. To achieve this, a joint training of an image encoder and a text encoder is performed. CLIP seeks to maximize the cosine similarity between the image and text for the N true pairs in the batch.

From the other hand qualitative methods, involve inspecting the generated data visually or auditorily. For example, you can use methods such as visual inspection, pairwise comparison, or preference ranking to assess how realistic, coherent, and appealing the generated data is. Qualitative methods can provide intuitive and subjective feedback on generative model performance, but they also have some drawbacks, such as being time-consuming, biased, or inconsistent.

Several papers [58], [10] refer to the evaluation of text-to-image models using a multi-task benchmark. The proposed benchmark consists of prompts covering tasks across various applications. Human raters are asked to select the best-generated images from the evaluated models based on image fidelity and image-text alignment. The evaluation method involves questions posed to human raters, inquiring about these two criteria.

Evaluating generative models is a complex task, as there are many challenges and trade-offs to consider. Choosing the right evaluation method and metric is essential for the specific generative model and task.

# 3. ASSESSING TEXT-TO-IMAGE MODELS

In this chapter, three text-to-image models for evaluation are presented, including the open-source models Stable Diffusion and Pix-Art-α, alongside the commercial model DALL-E 2. These models share three fundamental hyperparameters: prompt, height, and width. Additionally, a multi-task benchmark is defined upon which the models operate. The benchmark comprises six distinct task types, each encompassing twelve unique prompts. Two assessment techniques are used to evaluate the models. A qualitative analysis, where raters are presented with images generated by the models and asked to assess the alignment between the image and the accompanying text on a scale ranging from 1 (poor alignment) to 5 (excellent alignment). Subsequently, a quantitative evaluation is executed utilizing the CLIP model. This evaluation involves calculating similarity scores between each prompt and the corresponding generated image, providing an objective measure of performance.

## 3.1 Assessing Models

Exploring the behaviour of AI models like DALL-E 2, Stable Diffusion, and Pix-Art-α provides the opportunity to understand what these models are capable of.

### DALL-E 2

DALL-E 2 [64] is an artificial intelligence model that takes a text prompt as an input and generates a new image as an output. DALL-E 2 was developed by researchers at OpenAI to understand the capabilities and broader implications of multi-modal generative models. It can create more realistic images than DALL-E 1 at higher resolutions and can combine concepts, attributes, and styles. DALL-E 2 builds on DALL-E 1 increasing the level of resolution, fidelity, and overall photorealism it is capable of producing. DALL-E 2 is also trained to have new capabilities compared to DALL-E 1. Python code snippet has been written based on OpenAI API [1], which is available for download from git-hub.[1]

```
# imports

# OpenAI Python library to make API calls
from openai import OpenAI, OpenAIError
# Used to download images
import requests
# Used to access filepaths
import os
# Used to name the filename of the image
```

[1]https://github.com/EMazarakis/Generative-AI-in-Graphics/blob/main/DALL-E%202/PythonApplication5.py

```
import datetime
# Used to print and edit images
from PIL import Image

# Initialize OpenAI client
client = OpenAI(api_key=os.environ.get("OPENAI_API_KEY", "YOUR-API-KEY-HERE"))

# Set a directory to save DALL·E images to
image_dir_name = "C:/xxx/yyy/zzz/DALLE_IMAGES/"
image_dir = os.path.join(os.curdir, image_dir_name)

# Type of saved image
suffix = '.png'
# Supported sizes for 256x256, 512x512, or 1024x1024
MODEL = "dall-e-2"

# Custom exception class for OpenAI image errors
class OpenAIImageError(Exception):
    pass

# Function to get user input (prompt, size, number of images)
def get_user_input():
    prompt = input("Enter the prompt: ")
    size_option = input("Enter 1 for 1024x1024, 2 for 512x512, or 3 for 256x256:  ")
    if size_option == '1':
        size = "1024x1024"
    elif size_option == '2':
        size = "512x512"
    elif size_option == '3':
        size = "256x256"
    else:
        print("Invalid option! Using default size 1024x1024")
        size = "1024x1024"
    num_images = int(input("Enter the number of images: "))
    return prompt, size, num_images


# Function to generate image using OpenAI API
def generate_image(prompt, num_images, size, format, m):
    # Must catch the errors.
    try:
        generation_response = client.images.generate(
            model = m,
            prompt = prompt,
            n = num_images,
            size = size,
            response_format=format,
        )
        return generation_response
    except OpenAIError as e:
        raise OpenAIImageError("OpenAI API Error: " + str(e))
    except Exception as e:
        raise OpenAIImageError("An unexpected error occurred: " + str(e))


# Function to save image data to files
def save_image(res, size, prefix, suffix):
     # Generate timestamp for unique filenames
```

```
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")

    # Response.data is a list of JSON objects with the urls of each generated image.
    for i, image in enumerate(res.data):
        # Construct filename
        generated_image_name = f'image_{size}_{timestamp}_{i}_{suffix}'
        generated_image_filepath = prefix + generated_image_name
       # Extract image URL from response
        generated_image_url = res.data[i].url
        # Download the image
        generated_image = requests.get(generated_image_url).content

        # Save image data to file
        print('Saving file ' + generated_image_filepath)
        with open(generated_image_filepath, "wb") as image_file:
            # Write the image to the file
            image_file.write(generated_image)



# Main function to run the program
def main():
    # Setting a loop, so the user can generate images indefinitely
    while True:
        try:
           # Ask user for the input
            prompt, size, num_images = get_user_input()
            # Create the images
            response = generate_image(prompt, num_images, size, "url", MODEL)
            # Save the images to local disk
            save_image(response, size, image_dir, suffix)
        except  OpenAIImageError as e:
            print("Error:", e)
            print("Exiting program.")
            break

# Entry point of the script
if __name__ == "__main__":
    main()
```

**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 [67].

We use 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+” . [67], [68]

In order to run the stable diffusion model, we installed the AutomaticGui1111, a web interface for Stable Diffusion [3]. The web interface uses the Stable-Diffusion-v1-5 checkpoint and through the interface we can run several times the model based on prompt, width and height. The installation

instructions of the model can be found on git-hub.[2]

**Pix-Art-α**
Pix-Art-α [12] is an open-source text-to-image generation model. It was trained by researchers at Huawei Noah's Ark Lab. This is a model that can be used to generate and modify images based on text prompts. Pix-Art-α adopts the Diffusion Transformer [57] as the base architecture and tailors the transformer blocks to handle the unique challenges of text-to-image tasks. Model achieves superior image generation quality while significantly reducing training costs and $CO_2$ emissions. Pix-Art-α achieves near-commercial application standards in image generation quality. The three core designs are the following:

1. Training strategy decomposition: Pix-art divides its training process into three stages, each targeting distinct data types.
   (a) Focuses on pixel dependency learning.
   (b) Centers on aligning text with images.
   (c) Emphasizes high-resolution and aesthetically pleasing image generation.
2. Efficient T2I Transformer: Employing the Diffusion Transformer [57] as the foundational architecture, the Transformer blocks are creatively customized to address the distinctive demands of T2I tasks.
3. High-informative data: Generate captions with high information density, leveraging the state-of-the-art vision-language model LLaVA. [45]

We use the PixArt-XL-2-1024-MS checkpoint [60] within the web application [59] to generate images for the purpose of human evaluation based on our benchmark.

## 3.2 Benchmarking Tasks

In this section, we provide a multi-task benchmark for evaluating text-to-image models. Our benchmark comprises a suite of tasks across multiple applications that capture a model's ability to handle different features of a text prompt. Our benchmark prompt is inspired by the DrawBench [69], which consists of 11 categories with approximately 200 text prompts. Our benchmark contains six task types [58] and each task type contains 12 prompts. Our benchmark with 72 prompts is as follows:

1. Colouring task: The prompt is used to examine the ability of the image generation system to generate images that have colouring concepts.
   (a) A red coloured car.
   (b) A black colored car.
   (c) A pink colored car.
   (d) A navy blue colored car.
   (e) A red car and a white sheep.
   (f) A blue bird and a brown bear.
   (g) A green apple and a black backpack.
   (h) A green cup and a blue cell phone.

[2] https://github.com/EMazarakis/Generative-AI-in-Graphics/blob/main/Stable%20Diffusion/Installation_Instructions.md

(i) A red pencil in a green cup on a blue table.
(j) An office with five desks and seven colorful chairs.
(k) An orange bird scaring a blue scarecrow with a red pirate hat.
(l) Two pink football balls and three green basketball balls on a bench.

2. Counting task: The prompt is used to examine the ability of the image generation system to generate images that have counting concepts.

(a) One tennis ball on the court.
(b) Three people crossing the street.
(c) Two monkeys eating banana.
(d) Bench in a park with two backpacks on it.
(e) Nine children doing a circle dance around a Christmas tree.
(f) An office desk with six laptops on it.
(g) Person holding four toy pyramids.
(h) Five photograph prints are hanging to dry in a dark room.
(i) Person carrying a stack of twelve books.
(j) Two people juggling ten balls together.
(k) Birthday cake with exactly twenty candles on it.
(l) Office room with fifteen chairs.

3. Conflicting task: The prompt is used to examine the ability of the image generation system to generate images that have conflicting concepts.

(a) A zebra without strips.
(b) A penguin in a city.
(c) Rainbow colored Ferrari.
(d) A giraffe kissing a monkey.
(e) A panda inside a cup of tea.
(f) A giraffe inside a car.
(g) An ant under the sea.
(h) A motorcycle inside an oven.
(i) A polar bear on the desert.
(j) A man walking on the ceiling.
(k) A fish eating a pelican.
(l) A horse riding an astronaut in the forest.

4. Text task: The prompt is used to examine the ability of the image generation system to generate images that have text concepts.

(a) A sign that says 'Hello'.
(b) A sign that says 'World'.
(c) A sign that says 'Hello World'.
(d) A sign that says 'World Hello'.
(e) A sign that says 'Speed limit 45'.
(f) A sign that says 'Don't pass !'.
(g) A sign that says 'No pedestrians !'.
(h) A sign that says 'No overtaking !'.
(i) A sign that says 'No Goods materials'.
(j) A sign that says 'No hazardous materials'.
(k) A sign that says 'You must not turn Right!'.

(l) A sign that says 'Maximum speed limit 80 km'.

5. Positional task: The prompt is used to examine the ability of image generation systems to generate images with accurate positional information.
   (a) A train on top of a surfboard.
   (b) A wine glass on top of a dog.
   (c) A bicycle on top of a boat.
   (d) An umbrella under a spoon.
   (e) A car on the left of a bus.
   (f) A black apple on the right of a green backpack.
   (g) A carrot on the left of a broccoli.
   (h) A pizza on the right of a suitcase.
   (i) A cat on the right of a tennis racket.
   (j) A stop sign on the right of a refrigerator.
   (k) A sheep to the right of a wine glass.
   (l) A zebra to the right of a fire hydrant.
6. Faces task: The prompt is used to examine the ability of image generation systems to generate images with accurate facing information.
   (a) Face of a man with a goatee.
   (b) Face of an angry teacher.
   (c) Face of a kid with a tiger make-up.
   (d) Face of an old lady with blue hair and a wide smile.
   (e) A bald man doing a handstand.
   (f) Face of a woman with brown hair looking over her shoulder.
   (g) Sad face of a blonde girl holding a popped balloon.
   (h) Face of a weightlifting white hair Olympic gold medalist lifting 120kg with a referee next to it encouraging him.
   (i) The face of a soldier with camouflaged face painting obscured by leaves.
   (j) A face of a man with a beard with water splashing onto his face.
   (k) A dark messy hair boy rolling his tongue with blue light shinning on his face.
   (l) A young kid with dark eye bags standing in front of her grandma with wrinkles looking at the sky.

The file containing the prompts for the multi-task benchmark can be found on git-hub.[3]

## 3.3 Assessment Techniques

There are two methods for evaluating generative AI models. The first approach is a qualitative method that entails examining the generated data through human evaluation. Alternatively, the other approach involves quantitative method, where numerical scores are calculated based on certain criteria. For instance, metrics like CLIP score can be utilized for this purpose.

**Qualitative assessment**
Qualitative assessment of generated images entails a human evaluation. The quality is assessed

[3]https://github.com/EMazarakis/Generative-AI-in-Graphics/tree/main/Benchmarking

based on various criteria, including color, object count, conflicting scenarios, presence and quality of text, facial appearance, and spatial relationships. Our evaluation protocol consists of human ratings [58], [10]. For evaluation on the benchmark, we conduct an independent human evaluation run for each model, size, task type. For each prompt, the rater is shown images which is from DALL-E 2, from Stable Diffusion, and from Pix-Art-α. The human rater asked a single question and prompted to select a rating between 1 (worst) and 5 (best) [58] to indicate the alignment between the image and text. The question was:

1. How well does the image represent the text caption: [Text Caption] ? [10]
   (a) Question subjectively evaluates image-text alignment.

Examples of cases in which human evaluation is required are:

1. cases where objects with specified colors are generated (i.e. A pink coloured car).
2. cases where a specified number of objects needs to be generated (i.e. Two people juggling ten balls together).
3. concepts that are difficult to define, such as objects that rarely co-occur in the real world (i.e. A polar bear on the desert).
4. cases where aligning text according to the prompt's specifications (i.e. A sign that says 'Hello').
5. concepts where generating objects with specific spatial positioning is required (i.e. A zebra to the right of a fire hydrant).
6. concepts where generating realistic faces with or without a specific emotion is required (i.e. Face of an angry teacher).

We aggregate scores from 17 raters, the evaluated images were 432 (totalling to 17 raters × 3 models × 2 sizes × 6 tasks × 12 prompts = 7344 ratings). We do not perform any post filtering of the data to identify unreliable raters, both for expedience and because the task was straightforward to explain and execute. The scores from different raters aggregated and then converted into a normalized percentage value. Two questionnaires were distributed and the images were organized based on sizes: 512 [4] is the one and 1024 [5] is the other.

**Quantitative assessment**
The quantitative evaluation of generated images relies on metric scores. CLIP Score is a widely recognized method for measuring the similarity between an AI-generated image and its corresponding text caption. It serves as a potent tool for tasks in computer vision and language understanding. The primary objective of CLIP is to facilitate models in comprehending the correlation between visual and textual data.

CLIP [62], which stands for Contrastive Language-Image Pre-training, is a deep learning model developed by OpenAI in 2021. CLIP is designed to predict which N × N potential (image, text) pairings within the batch are actual matches. To achieve this, CLIP establishes a multi-modal embedding space through the joint training of an image encoder and text encoder. The CLIP loss aims to maximize the cosine similarity between the image and text embeddings for the N genuine

[4] https://docs.google.com/forms/d/e/1FAIpQLSfc6eiUOoyYl3_iSHRzdCexHwL_vRwRgiUb-94TF7aCI24jMg/viewform?usp=sf_link

[5] https://docs.google.com/forms/d/e/1FAIpQLSfLgMkBcQn4YpLN2Fq38EDBQGkwoDbjTliYvqQovoSGA-HtRw/viewform?usp=sf_link

pairs in the batch while minimizing the cosine similarity for the $N^2 - N$ incorrect pairings.

In CLIP, there are two main components, an image encoder and text encoder.

1. Image Encoder: represents the neural network responsible for encoding images. There are two different architectures utilized in image encoder:
   (a) ResNet
   (b) Vision Transformer
2. Text Encoder: represents the neural network responsible for encoding textual information. The architecture utilized by the text encoder in CLIP is the Transformer architecture.

The trained dataset of the CLIP model has 400 million (image, text) pairs collected form a variety of publicly available sources on the Internet. They refer to this dataset as WIT for WebImageText. The model was trained from scratch without initializing the image encoder and the text encoder with pre-trained weights due to the large volume of the dataset.

To assess the similarity between an image and a text description, we calculate the CLIP score, which represents the cosine similarity of the embeddings generated by the model for the image and text. To accomplish this, a Python code snippet has been written based on the official repo of openai [54] and is available for download from git-hub. [6]

```
# First you need to install the following :
# pip install -U torch torchvision
# pip install -U git+https://github.com/openai/CLIP.git

# Import all the appropriate libraries.
from pickle import FLOAT
import torch
import clip
from PIL import Image   # PIL: support opening, manipulating, and saving  image file formats
from torchvision.transforms import RandomEqualize
import csv
import time
import numpy as np
from collections import defaultdict


# If you want to see all the available models.
available_models = clip.available_models()
print("All the available models of CLIP family are: ", available_models)

#############################################################################
# All the relevant variables for the image file name.
#############################################################################
# Here is the directory prefix where the model images are hosted.
prefix_path = "C:/Users/XXX/YYY/000.Produced_Images/"

# The folder name for each model's images.
prefix_models = ["001.Stable_Diffusion", "002.DALLE_2", "003.Pix_Art_a"]

```

[6] https:/github.com/EMazarakis/Generative-AI-in-Graphics/tree/main/CLIP

```
# The folder name containing images of the appropriate size.
prefix_size = ["001.512_512","002.1024_1024"]

# The folder name of eash taksk type
prefix_task = ["001.Colouring", "002.Counting", "003.Conflicting", "004.Text", "005.
    Positional", "006.Faces"]

# Abbreviations of the model names
models_abbr = ["SD","DL", "PA"]

# The size (width, height) of each image
img_sizes = ["512", "1024"]

# The name of each task type of the benchmark
tasks = ["coloring", "counting", "conflicting", "text", "positional", "faces"]

# The number of images for each task type that we check
NUM_OF_IMAGES = 12


##############################################################################
# Our benchmark
##############################################################################
# Create a tuple of triples with three values each
my_benchmark_tuple = (
    ("coloring", "A red colored car", 1),
    ("coloring", "A black colored car", 2),
    ("coloring", "A pink colored car", 3),
    ("coloring", "A navy blue colored car", 4),
    ("coloring", "A red car and a white sheep", 5),
    ("coloring", "A blue bird and a brown bear", 6),
    ("coloring", "A green apple and a black backpack", 7),
    ("coloring", "A green cup and a blue cell phone", 8),
    ("coloring", "A red pencil in a green cup on a blue table", 9),
    ("coloring", "An office with five desks and seven colorful chairs", 10),
    ("coloring", "An orange bird scaring a blue scarecrow with a red pirate hat", 11),
    ("coloring", "A photo of two pink football balls and three green basketball balls
    on a bench", 12),
    ("counting", "One tennis ball on the court", 1),
    ("counting", "Three people crossing  the street", 2),
    ("counting", "Two monkeys eating banana", 3),
    ("counting", "Bench in a park with two backpacks on it", 4),
    ("counting", "Nine children doing a circle dance around a Christmas tree", 5),
    ("counting", "An office desk with six laptops on it", 6),
    ("counting", "Person holding  four toy pyramids", 7),
    ("counting", "Five photograph prints are hanging to dry in a dark room", 8),
    ("counting", "Person carrying a stack of twelve books", 9),
    ("counting", "Two people juggling ten balls together", 10),
    ("counting", "Birthday cake with exactly twenty candles on it", 11),
    ("counting", "Office room with fifteen chairs", 12),
    ("conflicting", "A zebra without strips", 1),
    ("conflicting", "A penguin in a city", 2),
    ("conflicting", "Rainbow colored Ferrari", 3),
    ("conflicting", "A giraffe kissing a monkey", 4),
    ("conflicting", "A panda inside a cup of tea", 5),
    ("conflicting", "A giraffe inside a car", 6),
    ("conflicting", "An ant under the sea", 7),
    ("conflicting", "A motorcycle inside an oven", 8),
```

```
    ("conflicting", "A polar bear on the desert", 9),
    ("conflicting", "A man walking on the ceiling", 10),
    ("conflicting", "A fish eating a pelican", 11),
    ("conflicting", "A horse riding  an astronaut in the forest", 12),
    ("text", "A sign that says 'Hello'", 1),
    ("text", "A sign that says 'World'", 2),
    ("text", "A sign that says 'Hello World'", 3),
    ("text", "A sign that says 'World Hello'", 4),
    ("text", "A sign that says 'Speed limit 45'", 5),
    ("text", "A sign that says 'Do not pass !'", 6),
    ("text", "A sign that says 'No pedestrians !'", 7),
    ("text", "A sign that says 'No overtaking !'", 8),
    ("text", "A sign that says 'No Goods materials'", 9),
    ("text", "A sign that says 'No hazardous materials'", 10),
    ("text", "A sign that says 'You must not turn Right!'", 11),
    ("text", "A sign that says 'Maximum speed limit 80 km'", 12),
    ("positional", "A train on top of a surfboard", 1),
    ("positional", "A wine glass on top of a dog", 2),
    ("positional", "A bicycle on top of a boat", 3),
    ("positional", "An umbrella under a spoon", 4),
    ("positional", "A car on the left of a bus", 5),
    ("positional", "A black apple on the right of  a green backpack", 6),
    ("positional", "A carrot on the left of a broccoli", 7),
    ("positional", "A pizza on the right of a suitcase", 8),
    ("positional", "A cat on the right of a tennis racket", 9),
    ("positional", "A stop sign on the right of a refrigerator", 10),
    ("positional", "A sheep to the right of a wine glass", 11),
    ("positional", "A zebra to the right of a fire hydrant", 12),
    ("faces", "Face of a man with a goatee", 1),
    ("faces", "Face of an angry teacher", 2),
    ("faces", "Face of a kid with a tiger make-up", 3),
    ("faces", "Face of an old lady with blue hair and a wide smile", 4),
    ("faces", "A bald man doing a handstand", 5),
    ("faces", "Face of a woman with brown hair looking over her shoulder", 6),
    ("faces", "Sad face of a blonde girl holding a popped balloon", 7),
    ("faces", "Face of a weightlifting white hair Olympic gold medalist lifting 120kg
    with a referee next to it encouraging him", 8),
    ("faces", "The face of a soldier with camouflaged face painting obscured by leaves
    ", 9),
    ("faces", "A face of a man with a beard with water splashing onto his face", 10),
    ("faces", "A dark messy hair boy rolling his tongue with blue light shinning on
    his face", 11),
    ("faces", "A young kid with dark eye bags standing in front of her grandma with
    wrinkles looking at the sky", 12)
)

# Convert the tuple of triples into a dictionary with keys as tuples of (key, number)
my_benchmark_dict={(key, number):value for key, value, number in my_benchmark_tuple}

################################################################################

# This function calculate the clip score between an image and a prompt
def calculate_clip_score(path_to_image, prompt):
    """
    Parameters
    ----------
    path_to_image : str
        A string which is the path to the image file.
```

```
140
141     prompt : str
142         A string that we want to see as far as it is related to the image.
143
144     Returns
145     -------
146     similarity_score: float
147         computes the dot product between image and text.
148     """
149     # Load the pre-trained CLIP model and the image
150     # preprocess: A torchvision transform that converts a PIL image into a tensor that
        the returned model can take as its input
151     model, preprocess = clip.load('ViT-L/14@336px')
152     try:
153       # Opens and identifies the given image file & returns and Image object.
154         image = Image.open(path_to_image, "r")
155     except IOError as e:
156         # Handle IO errors (e.g., file not found, cannot read file)
157         print("An error occurred during the opening of the image file:", e)
158     except Exception as e:
159         # Handle other exceptions
160         print("An unexpected error occurred during the opening of the image file:", e)
161
162     # Preprocess the image and tokenize the text
163     # Convert a PIL image to tensor and then in this tensor add a dimension
164   # image_input is a torch.Tensor object
165     image_data = preprocess(image).unsqueeze(0)
166
167   # Returns a two-dimensional tensor containing the resulting tokens, the tokenized
      representation of given input string(s)
168     text_data = clip.tokenize([prompt])
169
170     # Move the inputs to GPU if available
171     device = "cuda" if torch.cuda.is_available() else "cpu"
172     image_data = image_data.to(device)   # Move image_input to specified device
173     text_data = text_data.to(device)         # Move text_input  to specified device
174     model = model.to(device)                  # Move CLIP model to specified device
175
176     # Generate embeddings for the image and text
177     # Forward propagate input data through the network without computing gradients,
      thus saving memory and computation.
178     with torch.no_grad():
179       # Given a batch of images, returns the image features encoded by the vision
      portion of the CLIP model.
180         image_attr = model.encode_image(image_data)
181         # Given a batch of text tokens, returns the text features encoded by the
      language portion of the CLIP model.
182         text_attr = model.encode_text(text_data)
183
184     # Normalize the features
185     # dim = -1: The operation will be performed along the last dimension of the tensor
      .
186     # keepdim = True: The output tensor will have the same number of dimensions as the
        input tensor, with one of the dimensions having a size of 1.
187     # .norm(): Computes the L2 (Frobenius norm /Euclidean norm) norm along the last
      dimension of a tensor while retaining the dimensionality of the tensor.
188     # Normalizes each feature vector, ensuring that each vector has unit length in the
        feature space
```

```
    image_attr = image_attr / image_attr.norm(p='fro', dim=-1, keepdim=True)
    # Normalizes each feature vector, ensuring that each vector has unit length in the feature space
    text_attr = text_attr / text_attr.norm(p='fro', dim=-1, keepdim=True)

    # Calculate the cosine similarity to get the CLIP score
    # matmul: matrix multiplication between tensors.
    # text_attr.T: Transpose the text_attr tensor
    # Returns a scalar value from the multiplication of tensors.
    similarity_score = torch.matmul(image_attr, text_attr.T).item()

    return similarity_score


# Main function to run the program
def main():

    # File path to write the scores
    score_file = "C:/Users/XXX/YYY/clip_scores_file.csv"
    # This file contains Mean, Median, Std
    metrics_file = "C:/Users/XXX/YYY/clip_metrics_file.csv"

    # We store all the values on a list as tuple of six elements
    image_text_values_list = []

    # Record the start time
    start_time = time.time()

    print("Calculate CLIP Score:")
    print("For: " + str(len(prefix_models)) + " text-to-image model(s).")
    print("For: " + str(len(img_sizes)) + " image size(s).")
    print("For: " + str(len(tasks)) + " task type(s).")
    print("For: " + str(NUM_OF_IMAGES) + " image(s).")
    print("Total #CLIP scores: " + str(len(prefix_models)*len(img_sizes)*len(tasks)*NUM_OF_IMAGES))


    # Write score data to the file
    with open(score_file, mode='w', newline='') as file:
        writer = csv.writer(file)
        # Write header of the file
        writer.writerow(['Model', 'Size', 'Task', 'Image', 'Prompt', 'CLIP_Score'])

        # This  loop creates file name of each image and the corresponding prompt
        for i in range( len(prefix_models) ):      # For the 3 models
            for j in range( len(img_sizes) ):       # For the 2 sizes (512, 1024)
                for k in range( len(tasks) ):        # For the 6 task type
                    for l in range( NUM_OF_IMAGES ):    # For the 12 images


                        image_file_path = prefix_path + prefix_models[i] + "/" + prefix_size[j] + "/" + prefix_task[k] + "/" + ("0" if (l+1) > 9 else "00") + str(l+1) + "." + models_abbr[i] + "_" + img_sizes[j] + "_" + tasks[k] + "_" + str(l+1) + ".png"

                        # For a certain task type we get the appropriate prompt
                        # Access values by key and number for the benchmark dictionary
                        text = tasks[k]+": "+my_benchmark_dict[(tasks[k], (l+1))]
```

```
                    # Call the function to calculate similarity score between image and text
                    CLIP_SCORE = calculate_clip_score(image_file_path,  text)

                    # Everything listed below is to be written in a file.
            # For which model do we compute the CLIP Score
                    model = models_abbr[i]

                    # For which image size do we compute the CLIP Score
                    size = img_sizes[j]

                    # For which task type do we compute the CLIP Score
                    task_type = tasks[k]

                    # For which image do we compute the CLIP Score
                    photo =("0" if (l+1) > 9 else "00")+str(l+1)+"."+models_abbr[i]+"_"+img_sizes[j]+"_"+tasks[k]+"_"+str(l+1)+".png"

                    # For which prompt do we compute the CLIP Score
                    prompt = my_benchmark_dict[(tasks[k], (l+1) )]

            # This is the requested value, truncated to 4 decimal digits
                    score =  format(CLIP_SCORE, '.4f')

                    # Define a tuple of six values
                    temp_tuple = (model, size, task_type, photo, prompt, score)

                    # Add the tuple to the list
                    image_text_values_list.append(temp_tuple)

                    # Write data row in the file
                    writer.writerow([model,size task_type,photo,prompt,score.replace('.', ',')])


    ###############################################################################
    # We are going to compute the Mean,Median and Std for each Task Type Per Model

    # Define the indices of values within each tuple
    model_index = 0
    size_index = 1
    category_index = 2
    value_index = 5

    # Initialize dictionaries to store means and medians for each combination/task
    means_dict = defaultdict(list)
    medians_dict = defaultdict(list)
    std_dict = defaultdict(list)

    for model, size, category, _, _, value in image_text_values_list:
        key = (model, size, category)
        means_dict[key].append(float(value))
        medians_dict[key].append(float(value))
        std_dict[key].append(float(value))

    # Calculate means and medians for each combination
    means_result = {}
```

```
    medians_result = {}
    std_result = {}

    for key, values in means_dict.items():
        means_result[key] = np.mean(values)

    for key, values in medians_dict.items():
        medians_result[key] = np.median(values)

    for key, values in std_dict.items():
        std_result[key] = np.std(values)

    # Write metrics data to the file
    with open(metrics_file, mode='w', newline='') as file:
        metrics_writer = csv.writer(file)
        # Write the header of the file
        metrics_writer.writerow(['Model','Size','Task','Mean','Mean%','Median','Median%','Std','Std%'])

        # Print means and medians for each combination
        for key in means_result:
            #Model:key[0], Size: key[1], Category: key[2]
            # Write data row in the file
            metrics_writer.writerow([ key[0], key[1], key[2], format(means_result[key], '.4f').replace('.', ','), format(means_result[key]*100, '.4f').replace('.', ','), format(medians_result[key], '.4f').replace('.', ','), format(medians_result[key]*100, '.4f').replace('.', ','), format(std_result[key], '.4f').replace('.', ','), format(std_result[key]*100, '.4f').replace('.', ',') ])

    # Record the end time
    end_time = time.time()

    # Calculate the duration
    duration = (end_time - start_time)/60
    print("--------------------------------------------------------------------------")
    print("CLIP scores computed on:  ", duration, "minutes")
    print("Scores has been written to ", score_file)
    print("Metrics has been written to ", metrics_file)
    print("--------------------------------------------------------------------------")


# Entry point of the script
if __name__ == "__main__":
    main()
```

# 4. APPRAISAL OUTCOMES

In this chapter, we present the results of the qualitative and quantitative assessment of the three text-to-image models (DALL-E 2, Stable Diffusion, Pix-art-α). Additionally, we discuss observations based on the generated images of the models.

## 4.1 Results

**Qualitative assessment**
We surveyed 17 people (employees from the Data and Analytics department of IT companies) who evaluated the performance of three image generation models: Stable Diffusion, DALL-E 2, and Pix-Art-α. Each person evaluated 432 images, providing ratings between 1 (worst) and 5 (best). We collected a total of 7344 scores from 2 questionnaires, 512 and 1024. We use the min-max normalization technique to scale the values of the human evaluation to a specific range [0, 100]. The formula that was used is:

$$\frac{\text{EvaluationValue} - \text{MinEvaluationValue}}{\text{MaxEvaluationValue} - \text{MinEvaluationValue}}$$

The result of the formula was converted to a percentage by multiplying it by 100 to obtain the normalized ratings (%) for human evaluation and categorizing the data by model, size, and task type. This allowed us to analyze and compare the performance, providing deeper insights into the effectiveness and efficiency of each model variation. The analysis of the human evaluation scores was performed using power bi desktop app. Data transformations were applied to achieve the desired format. All relevant transformations can be found on github. [1] The normalized scores for images of size (width, height) = (512, 512) are presented in figure 4.1. Similarly, the scores for images of size (width, height) = (1024, 1024) are presented in figures 4.2. All the human evaluation scores are available on github. [2]

If we examine the charts depicting the score values for each task across the models, specifically for images sized 512x512 (figure 4.1), we can make several observations:

1. DALL-E 2: Raters identified four task categories where this model performed well.
   (a) DALL-E 2 achieved the highest rater scores for image-text alignment in coloring, counting, faces, and text tasks. This suggests that images generated by DALL-E 2 best

[1] https://github.com/EMazarakis/Generative-AI-in-Graphics/blob/main/Human%20Evaluation/Power%20BI%20Analysis/PBI%20Transformations.md
[2] https://github.com/EMazarakis/Generative-AI-in-Graphics/tree/main/Human%20Evaluation/Questionnaire%20Data

Normalized ratings (%) for human evaluation by Task and Model

| Task Type | DALL-E 2 | Pix-Art-α | Stable Diffusion |
|---|---|---|---|
| Colouring | 74,51 | 67,28 | 60,17 |
| Conflicting | 49,75 | 61,76 | 34,44 |
| Counting | 74,02 | 51,47 | 41,91 |
| Faces | 74,02 | 69,12 | 52,08 |
| Positional | 41,30 | 48,53 | 40,44 |
| Text | 38,60 | 13,24 | 13,73 |

Figure 4.1: Normalized ratings (%) for human evaluation and images sized (512,512). Highlight the maximum value using the color red.

match the descriptions provided in the prompts for these aspects (coloring, counting, faces, and text).

2. Pix-Art-α: Raters identified two task categories where this model performed well.
   (a) Pix-Art-α achieved the highest rater scores for image-text alignment in conflicting and positional tasks. This suggests that images generated by Pix-Art-α best match the descriptions provided in the prompts for these aspects (conflicting, positional).
3. Stable Diffusion: Raters did not identify any single task category where this model performed well.

Normalized ratings (%) for human evaluation by Task and Model

| Task Type | DALL-E 2 | Pix-Art-α | Stable Diffusion |
|---|---|---|---|
| Colouring | 72,06 | 64,22 | 31,74 |
| Conflicting | 50,00 | 58,46 | 27,33 |
| Counting | 71,08 | 56,37 | 31,37 |
| Faces | 75,49 | 72,06 | 27,08 |
| Positional | 46,94 | 42,28 | 25,37 |
| Text | 49,02 | 15,44 | 13,11 |

Figure 4.2: Normalized ratings (%) for human evaluation and images sized (1024,1024). Highlight the maximum value using the color red.

If we examine the charts depicting the score values for each task across the models, specifically for images sized 1024x1024 (figure 4.2), we can make several observations:

1. DALL-E 2: Raters identified five task categories where this model performed well.
   (a) DALL-E 2 achieved the highest rater scores for image-text alignment in coloring, counting, faces, positional and text tasks. This suggests that images generated by DALL-E 2 best match the descriptions provided in the prompts for these aspects (coloring, counting, faces, positional and text).
2. Pix-Art-α: Raters identified one task category where this model performed well.

(a) Pix-Art-α achieved the highest rater scores for image-text alignment in conflicting task. This suggests that images generated by Pix-Art-α best match the descriptions provided in the prompts for these aspects (conflicting).

3. Stable Diffusion: Raters did not identify any single task category where this model performed well.

(a) The model's ratings are not only low, but also show a significant gap compared to the scores of the previous model.

**Quantitative assessment**

We ran the ViT-L/14@336px model [62] , which demonstrates superior performance compared to all other models within the CLIP family. This model utilizes the Vision Transformer as its image encoder. We compute the clip score between the prompt and image to ascertain the image-text alignment similarity score. Additionally, we calculated the mean and median percentages of the clip scores, categorizing the data by model, size, and task type. This allowed us to analyze and compare the performance, providing deeper insights into the effectiveness and efficiency of each model variation. Detailed CLIP scores and statistical metrics are available on github. [3]

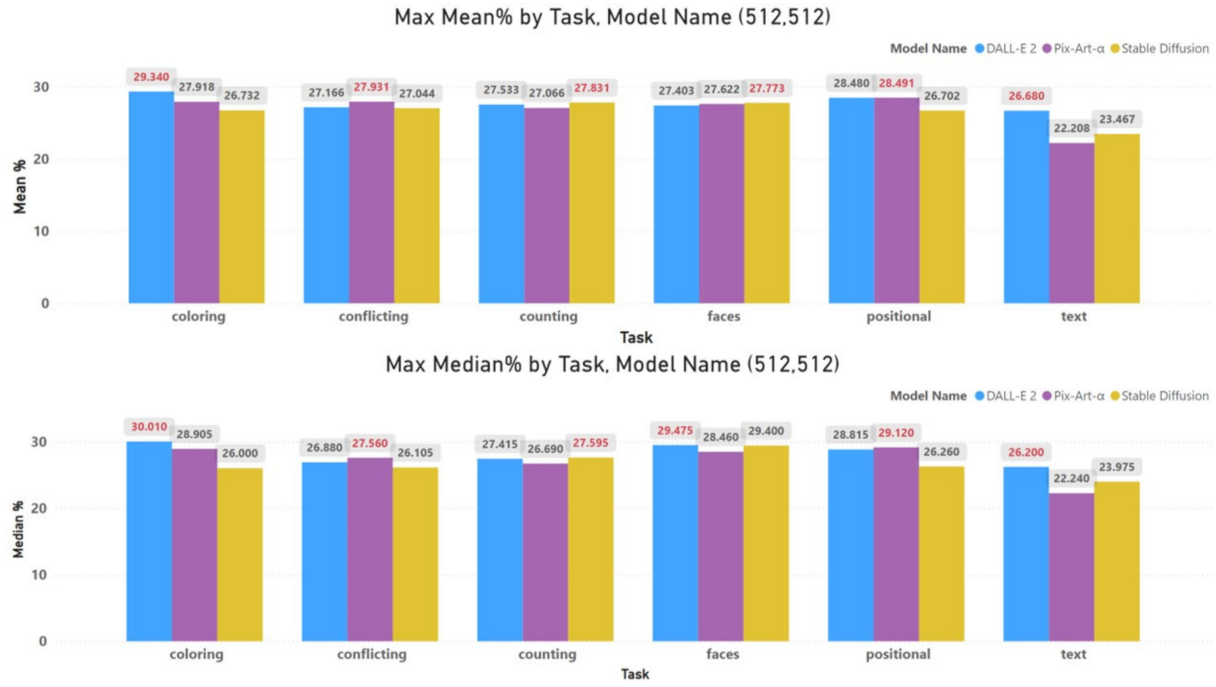


Figure 4.3: Mean% and Median% for images (512,512) by Task, Model Name based on ViT-L/14@336px.Highlight the maximum value within each cluster using the color red.

If we examine the chart depicting the mean and median values for each task across the models, specifically for images sized 512x512, we can make several observations. (see Figure 4.3) The results of image-text alignment for each model based on CLIP score are:

[3]https://github.com/EMazarakis/Generative-AI-in-Graphics/tree/main/CLIP%20Score

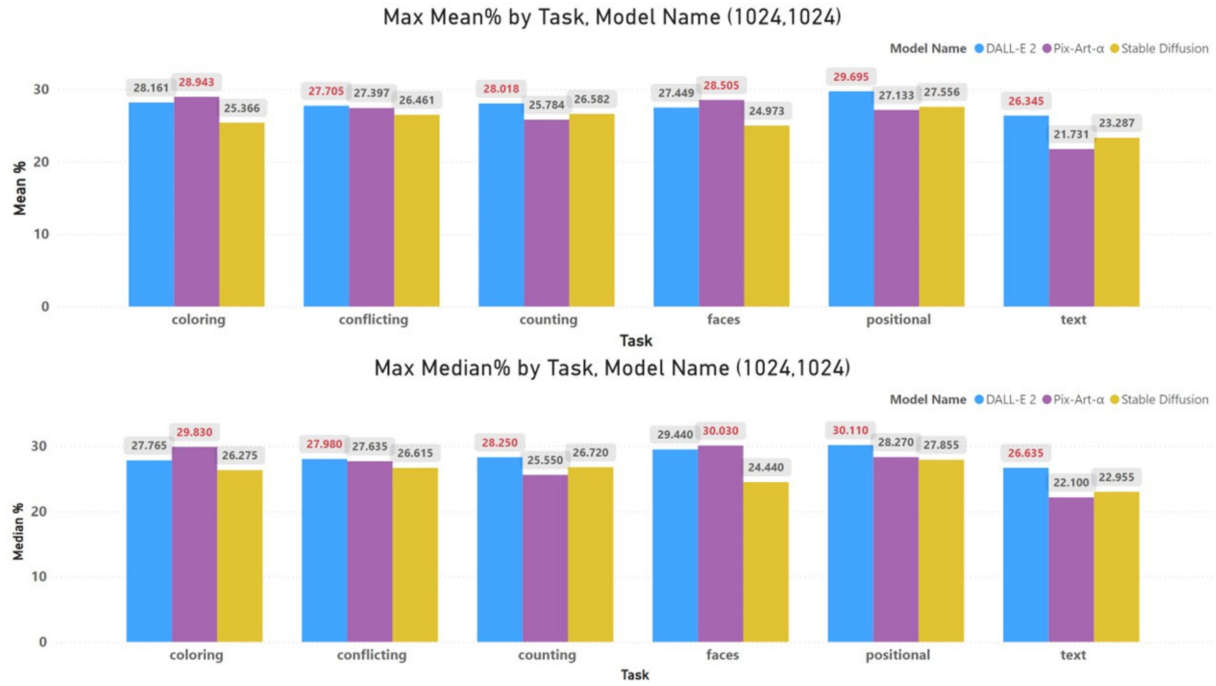


Figure 4.4: Mean% and Median% for images (1024, 1024) by Task, Model Name based on ViT-L/14@336px.Highlight the maximum value within each cluster using the color red.

1. Stable Diffusion: counting
   (a) Stable Diffusion effectively translates the count descriptions into visually accurate images.
2. DALL-E 2: coloring, text, faces
   (a) DALL-E 2 has the best image-text alignment CLIP score on coloring and text tasks, suggesting that its generated images closely align with the given prompts in terms of color and demonstrating its proficiency in translating textual prompts into visually coherent images.
   (b) On the faces task, DALL-E 2's higher median suggests it performs more consistently well across the dataset, while Stable Diffusion's higher mean indicates that it has some very high-performing instances that raise its average score. This indicates that DALL-E 2 is better in generating images with facial features.
3. Pix-Art-α: conflicting, positional
   (a) Pix-Art-α is particularly proficient in generating images based on positional prompts and conflicting scenarios.

If we examine the chart depicting the mean and median values for each task across the models, specifically for images sized 1024x1024, we can make several observations. (see Figure 4.4) The results of image-text alignment for each model based on CLIP score are:

1. Stable Diffusion: N/A

   (a) The stable diffusion does not achieve a maximum value on any task. This is to be expected, since the images are not suitable for any category.(see figure 4.8)

2. DALL-E 2: conflicting, counting, positional, text
   (a) DALL-E 2 achieved the highest scores for conflicting scenarios, count descriptions, positional prompts and textual prompts, indicating its proficiency in translating the prompts into visually coherent images.
3. Pix-Art-α: coloring, faces
   (a) Pix-Art-α has the best image-text alignment clip score on coloring and faces task, suggests that its generated images closely align with the given prompts in terms of color and in terms of facial characterisircs.

## 4.2 Discussion

**Stable Diffusion**
The Stable Diffusion model encounters challenges in effectively overlaying textual content onto images, manifesting in instances of poor alignment or incoherence, thereby compromising visual integrity. Additionally, it exhibits difficulty in synthesizing human representations within images, leading to depictions lacking in verisimilitude and authentic human traits. Moreover, it faces obstacles in generating images characterized by intricate compositions or tasks, often producing disjointed or implausible scenes due to its limited capacity to seamlessly integrate diverse visual elements. The aforementioned observations align with the discoveries documented in the models' card. [16] These limitations underscore the model's ongoing developmental requirements for enhancing text-image integration, human likeness depiction, and compositional complexity generation. The main shortcomings or limitations present in the generative model arise when we opt for the hyperparameter values of **(width, height) = (512,512)** , which consist of:

1. Legible text cannot be rendered by the model. [16] (see figure 4.5)
   (a) prt: A sign that says 'Hello'.
   (b) prt: A sign that says 'No hazardous materials'.
   (c) prt: A sign that says 'Maximum speed limit 80 km'.
2. People characteristics are not generated properly. [16] (see figure 4.6)
   (a) Faces are not displayed correctly.
      i. prt: Two people juggling ten balls together.
   (b) The fingers of the hands are not displayed correctly.
      i. prt: Sad face of a blonde girl holding a popped balloon.
   (c) The pupils of the eyes are not depicted correctly.
      i. prt: Face of a kid with a tiger make-up.
3. Difficult tasks involving compositionality are not generated properly. [16] (see figure 4.7)
   (a) prt: A carrot on the left of a broccoli.
   (b) prt: A zebra to the right of a fire hydrant.
   (c) prt: An umbrella under a spoon.

The main issue in the generative model arises when opting for hyperparameter values **(width, height) = (1024,1024)** , resulting in peculiar behaviour. The performance of the model drastically

changes when we use those values. While we anticipate images with enhanced resolution and attention to detail, the outcome is different. It appears that the model fails to comprehend the necessary adjustments, leading to repetitive patterns within the image, resulting in a bewildering output. It seems to iterate the input theme multiple times in its endeavour to populate the image with content. It struggles to create a coherent design or background, failing to acknowledge the shift in image size and redundantly overlaying the same pattern within the image. The aforementioned observation align with the discoveries documented in the models' card. [16]

The model's performance falls short across all six task types of the benchmark. (see figure 4.8)

1. prt: A pink colored car. (colouring task)
2. prt: Three people crossing the street. (counting task)
3. prt: A horse riding an astronaut in the forest. (conflicting task)
4. prt: A sign that says 'You must not turn Right!'. (text task)
5. prt: A zebra to the right of a fire hydrant. (positional task)
6. prt: A dark messy hair boy rolling his tongue with blue light shining on his face. (faces task)

The entire collection of generated images corresponding to Stable Diffusion and for the dimensions (512,512) and (1024,1024) across all benchmark prompts is accessible on github.[4]

**DALL-E 2**

The model encounters challenges in its functionality, including potential harms and implications, prompting us to draw conclusions. DALL-E 2 faces challenges in generating coherent text and also encounters difficulty in rendering details within complex scenes, it creates scenes with generic backgrounds. These observations are consistent with the findings presented in the paper 'Hierarchical Text-Conditional Image Generation with CLIP Latents' [64]. Furthermore, the model struggles to accurately generate human characteristics, particularly in representing facial features and hand movements.

The main shortcomings or limitations present in the generative model arise when we opt for the hyperparameter values to **(width, height) = (512,512)** , which consist of:

1. DALL-E 2 has difficulties on spelling. It predominantly generates English letters and arranges them in an order that more closely resembles English spelling than a random selection from a miscellaneous assortment of letters. The "words" on images are only approximations of the correct phrase. [64] (see figure 4.9)
   (a) prt: A sign that says 'Speed limit 45'.
   (b) prt: A sign that says 'No pedestrians !'.
   (c) prt: A sign that says 'You must not turn Right !'.
2. People characteristics are not generated properly.
   (a) Faces are not displayed correctly when they are not defined properly on the prompt. (see figure 4.10)
      i. prt: Two people juggling ten balls together.
      ii. prt: A man walking on the ceiling.
      iii. prt: A bald man doing a handstand.
   (b) The fingers of the hands are not displayed correctly. (see figure 4.11)

[4] https://github.com/EMazarakis/Generative-AI-in-Graphics/tree/main/Generative%20Images/SD

i. prt: Person holding four toy pyramids .
ii. prt: Two people juggling ten balls together.
iii. prt: Face of an angry teacher.

(c) The pupils of the eyes are not depicted correctly. (see figure 4.12)
i. prt: Face of a kid with a tiger make-up.
ii. prt: The face of a soldier with camouflaged face painting obscured by leaves.
iii. prt: Sad face of a blonde girl holding a popped balloon.

3. DALL-E 2 can do scenes with generic backgrounds (a city, a landscape, etc) but even then, if that's not the main focus of the image then the fine details tend to get pretty scrambled [64] (see figure 4.13)
(a) prt: Three people crossing the street.
(b) prt: An orange bird scaring a blue scarecrow with a red pirate hat.
(c) prt: A zebra to the right of a fire hydrant.

When we change the hyperparameter values to (width, height) = (1024, 1024), the main shortcomings or limitations remain:

1. DALL-E has difficulties on spelling. It becomes evident that DALL-E 2 exhibits improved performance in replicating text, but the difficulties on spelling remain. [64] (see figure 4.14)
(a) prt: A sign that says 'Speed limit 45'.
(b) prt: A sign that says 'No pedestrians !'.
(c) prt: A sign that says 'You must not turn Right!'.
2. People characteristics are not generated properly. It forgets how faces and fingers work.
(a) Faces are not displayed correctly when they are not defined properly on the prompt. (see figure 4.15)
i. prt: Two people juggling ten balls together.
ii. prt: A bald man doing a handstand.
iii. prt: Face of an angry teacher
(b) The fingers of the hands are not displayed correctly. (see figure 4.16)
i. prt: Person holding four toy pyramids.
ii. prt: Nine children doing a circle dance around a Christmas tree.
iii. prt: Person carrying a stack of twelve books.
(c) The pupils of the eyes are not depicted correctly. (see figure 4.17)
i. prt: Face of a kid with a tiger make-up.
ii. prt: The face of a soldier with camouflaged face painting obscured by leaves.
iii. prt: Sad face of a blonde girl holding a popped balloon.
3. DALL-E is capable of creating scenes with generic backgrounds, like a city or landscape. However, when these backgrounds are not the primary focus, the fine details often become quite jumbled. [64] (see figure 4.18)
(a) prt: Three people crossing the street.
(b) prt: Rainbow colored Ferrari.
(c) prt: A polar bear on the desert.

The entire collection of generated images corresponding to DALL-E 2 and for the dimensions (512,512) and (1024,1024) across all benchmark prompts is accessible on github. [5]

**Pix-Art-α**

The model demonstrates deficiencies and shortcomings in its image generation capabilities, revealing instances of failure and limitations. Pix-Art-α faces challenges in generating coherent textual content and accurately representing human characteristics, with a particular emphasis on facial features and hand movements. Furthermore, the model has weaknesses in accurately controlling the number of objects in an image. These difficulties are consistent with the conclusions elucidated in the paper "PIXART-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis" [12]. The primary deficiencies or constraints inherent in the generative model emerge when selecting the hyperparameter values for **(width, height) = (512,512)**, encompassing:

1. Pix art-a cannot render legible text. It does not accurately depict English alphabet characters, instead exclusively presenting symbols. Furthermore, it demonstrates incomplete understanding of the prompt. [12] (see figure 4.19)
   (a) prt: A sign that says 'Hello'.
   (b) prt: A sign that says 'World' .
   (c) prt: A sign that says 'No hazardous materials'.
2. People characteristics are not generated properly. It forgets how faces, hands and fingers work. The hands exhibit a greater length than anticipated, while the fingers appear as an amorphous mass devoid of defined shape. [12]
   (a) Faces are not displayed correctly when they are not defined properly on the prompt. (see figure 4.20)
      i. prt: Three people crossing the street.
      ii. prt: Nine children doing a circle dance around a Christmas tree.
      iii. prt: Two people juggling ten balls together.
   (b) The fingers and hands are not displayed correctly. (see figure 4.21)
      i. prt: Face of a weightlifting white hair Olympic gold medalist lifting 120kg with a referee next to it encouraging him.
      ii. prt: Person holding four toy pyramids.
      iii. prt: A bald man doing a handstand.
3. The model struggles with counting tasks. It has weaknesses in accurately controlling the number of objects in an image. [12] (see figure 4.22)
   (a) prt: Three people crossing the street.
   (b) prt: Nine children doing a circle dance around a Christmas tree.
   (c) prt: Two people juggling ten balls together.

Even though we changed the parameters **(width, height) = (1024,1024)** the model still struggles in some cases and does not perform well. The subsequent observations are in accordance with the findings documented in the paper "PIXART-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis" [12].

1. Pix art-a cannot render legible text. The representation of English alphabet characters is inaccurate, as it solely displays symbols. Moreover, it reveals an incomplete grasp of the given instructions. [12]

[5] https://github.com/EMazarakis/Generative-AI-in-Graphics/tree/main/Generative%20Images/DL

(a) The representation of English alphabet characters is inaccurate, as it solely displays symbols. Moreover, it reveals an incomplete grasp of the given instructions. (see figure 4.23)
   i. prt: A sign that says 'World'.
   ii. prt: A sign that says 'Speed limit 45'.
   iii. prt: A sign that says 'Maximum speed limit 80 km'.

(b) While it attempts to showcase some Latin alphabet characters, the text is primarily overrun by scattered symbols. (see figure 4.24)
   i. prt: A sign that says 'No pedestrians !'.
   ii. prt: A sign that says 'No Goods materials'.
   iii. prt: A sign that says 'No hazardous materials'.

2. People characteristics are not generated properly. The depiction fails to adhere to anatomical accuracy in portraying facial features, hands, and fingers. Notably, the hands are represented with disproportionate length, and the fingers lack discernible form, resembling a shapeless mass. [12]

(a) Faces are not displayed correctly when they are not defined properly on the prompt. (see figure 4.25)
   i. prt: Three people crossing the street.
   ii. prt: A bicycle on top of a boat.
   iii. prt: A bald man doing a handstand.

(b) The fingers and hands are not displayed correctly. (see figure 4.26)
   i. prt: Nine children doing a circle dance around a Christmas tree.
   ii. prt: Person holding four toy pyramids.
   iii. prt: Two people juggling ten balls together.

3. The model cannot count the objects in an image and struggles to accurately manage their number. (see figure 4.27)

(a) prt: Person carrying a stack of twelve books.
(b) prt: Two people juggling ten balls together.
(c) prt: An office desk with six laptops on it.

The entire collection of generated images corresponding to Pix-Art-α and for the dimensions (512,512) and (1024,1024) across all benchmark prompts is accessible on git-hub. [6]

[6] https://github.com/EMazarakis/Generative-AI-in-Graphics/tree/main/Generative%20Images/PA


Figure 4.5: Text limitations of the Stable Diffusion model for (width, height) = (512,512).


Figure 4.6: The limitations pertaining to people characteristics in the Stable Diffusion model for (width, height) = (512,512).

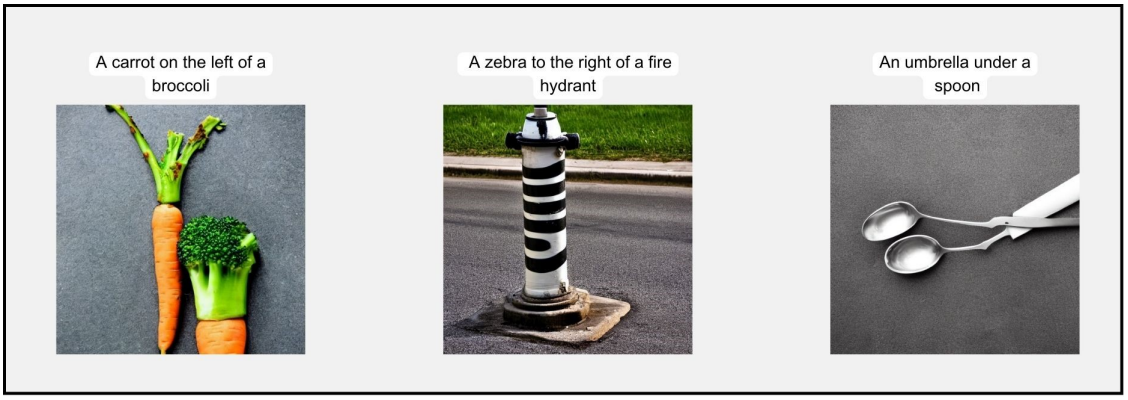


Figure 4.7: The limitations of the Stable Diffusion model for compositionality tasks at (width, height) = (512,512).

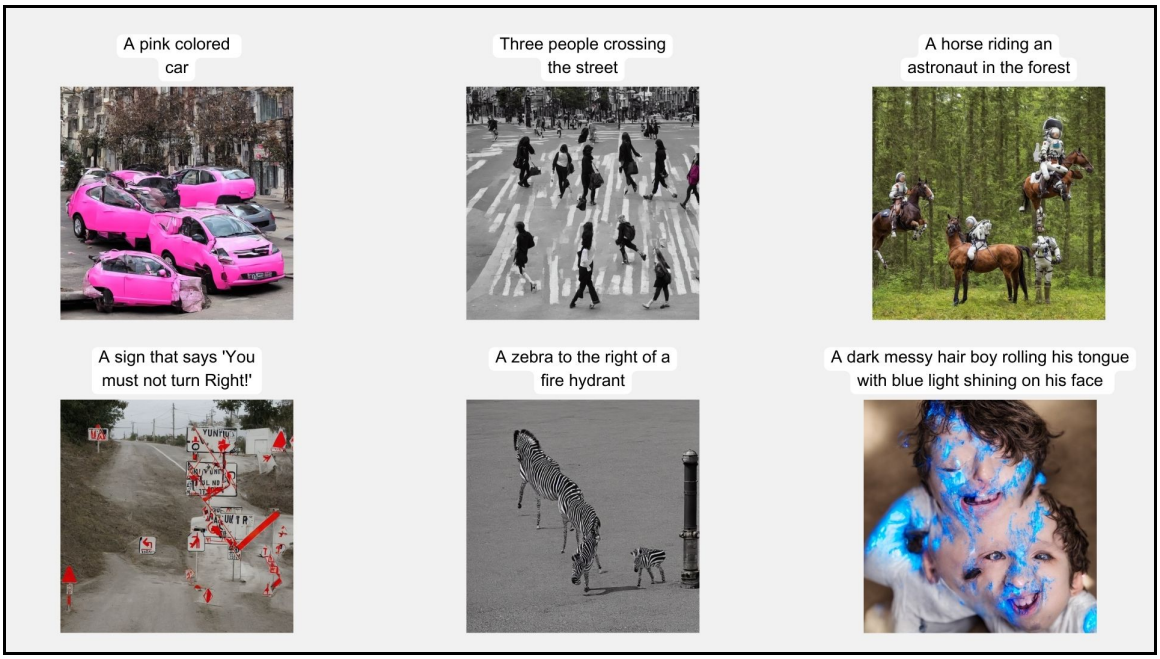


Figure 4.8: The Stable Diffusion model exhibits subpar performance across all six task types within the benchmark for (width, height) = (1024,1024).

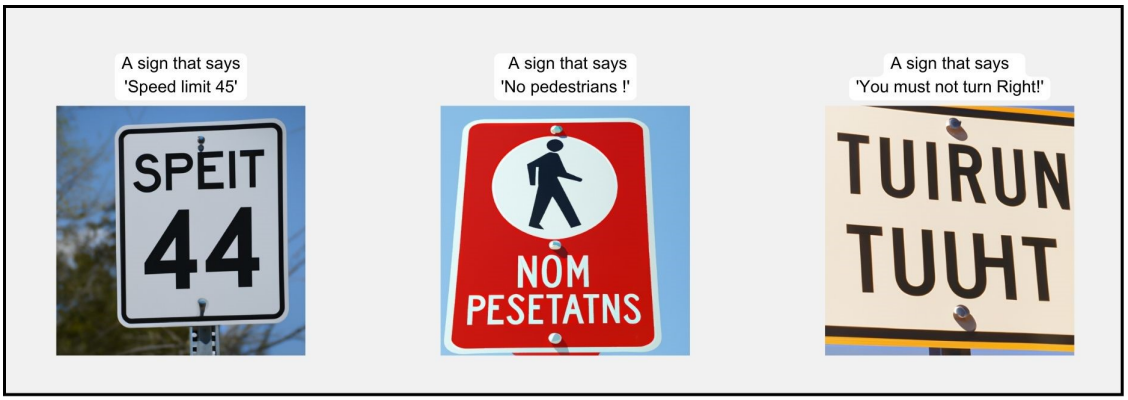


Figure 4.9: Text limitations of the DALL-E 2 model for (width, height) = (512,512).

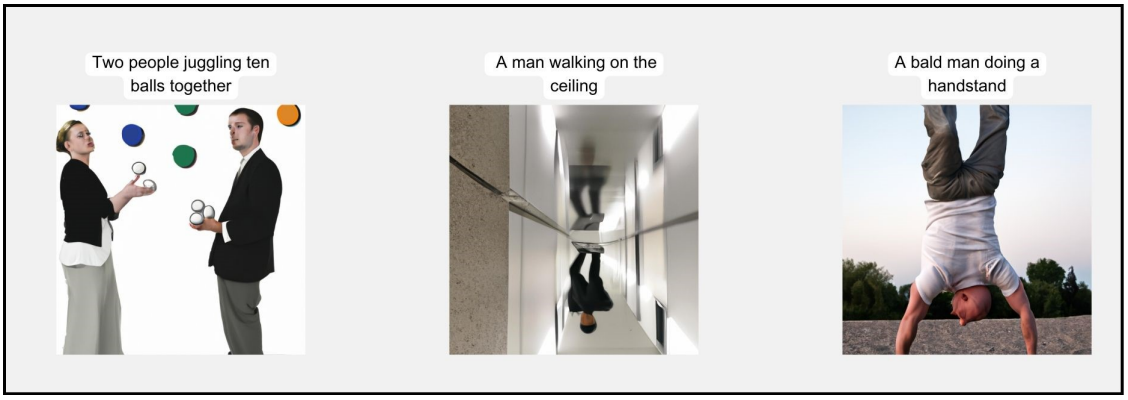


Figure 4.10: The limitations concerning characteristics of people's faces in the DALL-E 2 model for (width, height) = (512,512).

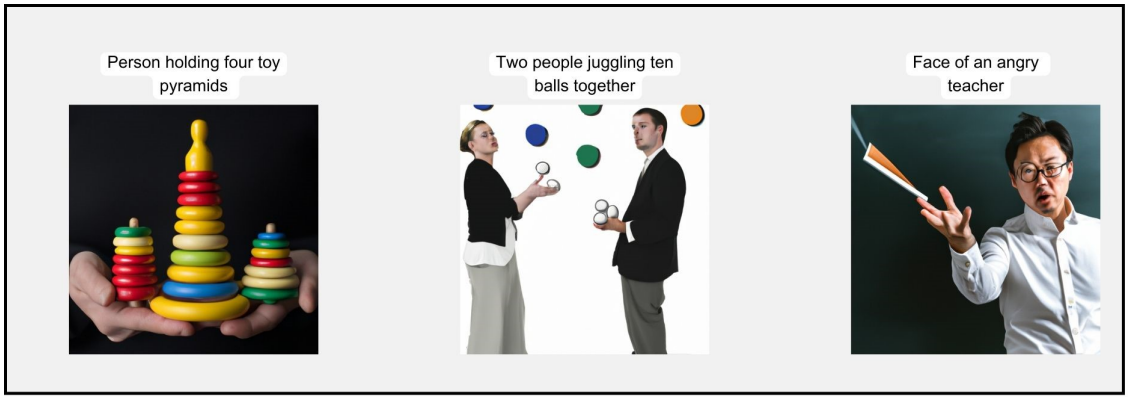


Figure 4.11: The limitations concerning characteristics of people's fingers in the DALL-E 2 model for (width, height) = (512,512).

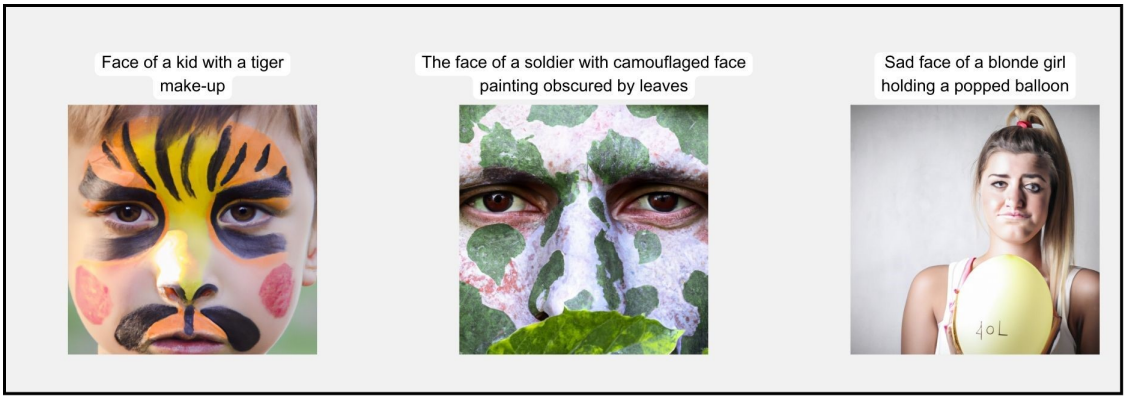


Figure 4.12: The limitations concerning characteristics of people's eye pupils in the DALL-E 2 model for (width, height) = (512,512).


Figure 4.13: The limitations regarding image background in the DALL-E 2 model for (width, height) = (512,512).


Figure 4.14: Adjusting the dimensions to (width, height) = (1024,1024) in the DALL-E 2 model results in improved spelling performance.

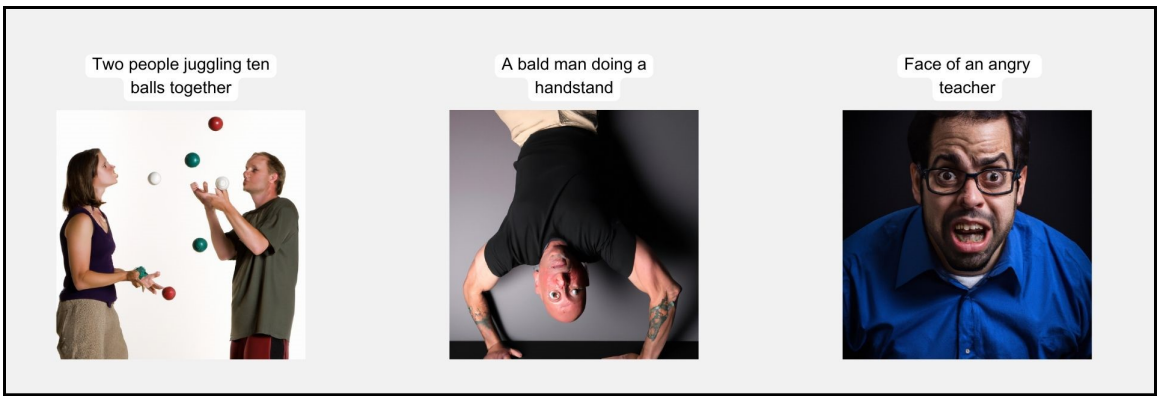


Figure 4.15: The limitations concerning characteristics of people's faces in the DALL-E 2 model for (width, height) = 1024,1024).


Figure 4.16: The limitations concerning characteristics of people's fingers in the DALL-E 2 model for (width, height) = (1024,1024).

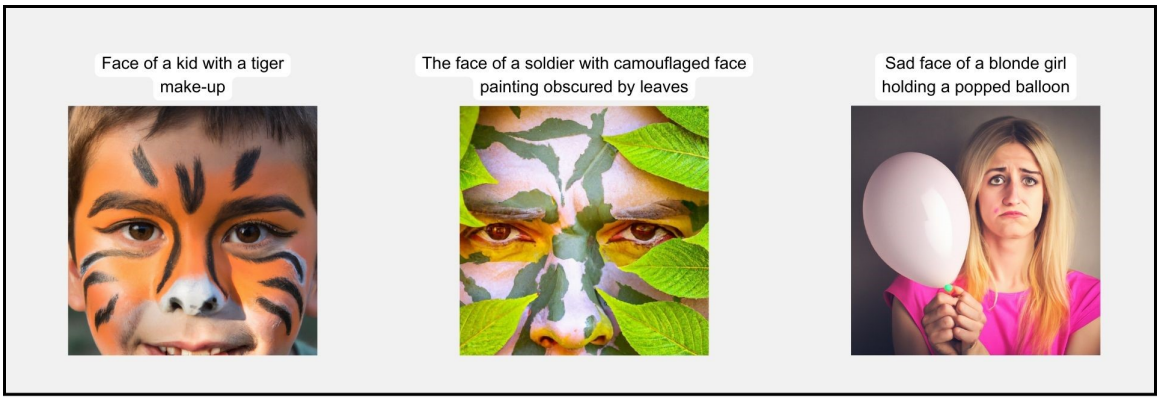


Figure 4.17: The limitations concerning characteristics of people's eye pupils in the DALL-E 2 model for (width, height) = (1024,1024).

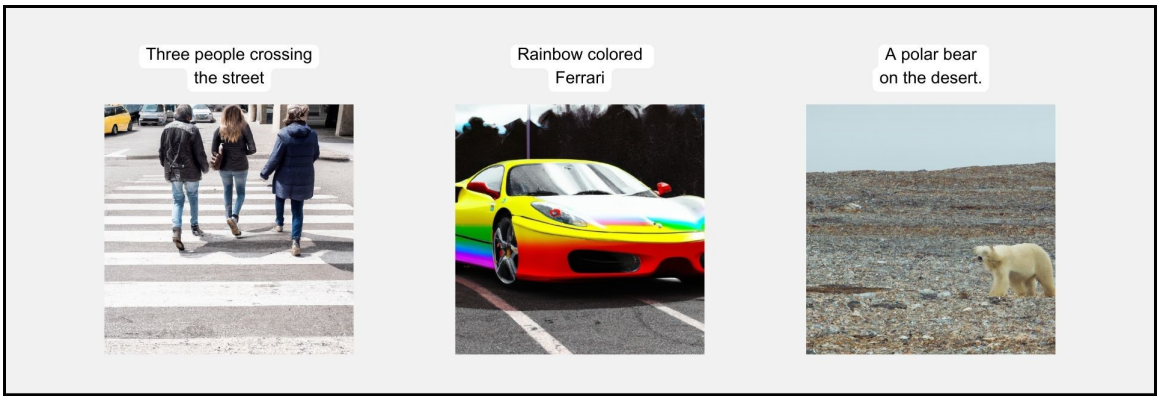


Figure 4.18: The limitations regarding image background in the DALL-E 2 model for (width, height) = (1024,1024).


Figure 4.19: Text limitations of the Pix-Art-α model for (width, height) = (512,512).


Figure 4.20: The limitations pertaining to people characteristics in the Pix-Art-α model for (width, height) = (512,512).

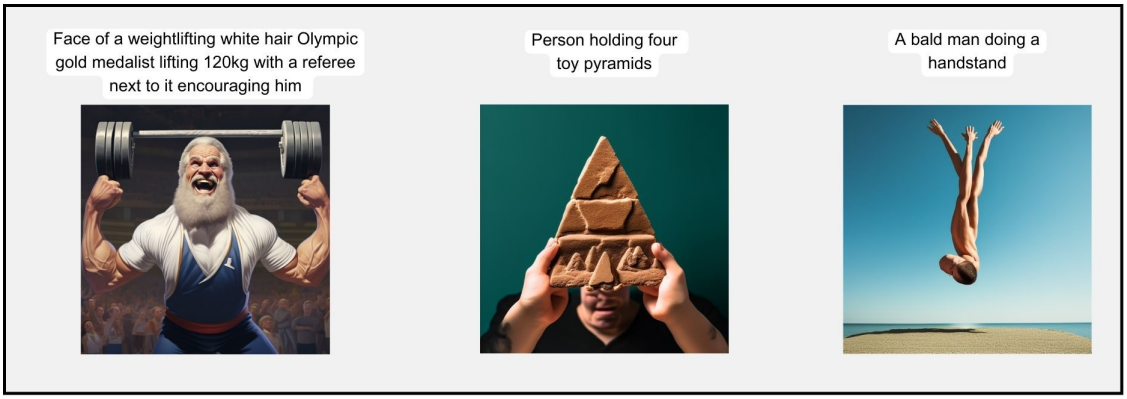


Figure 4.21: The limitations concerning characteristics of people's fingers and hands in the Pix-Art-α model for (width, height) = (512,512).

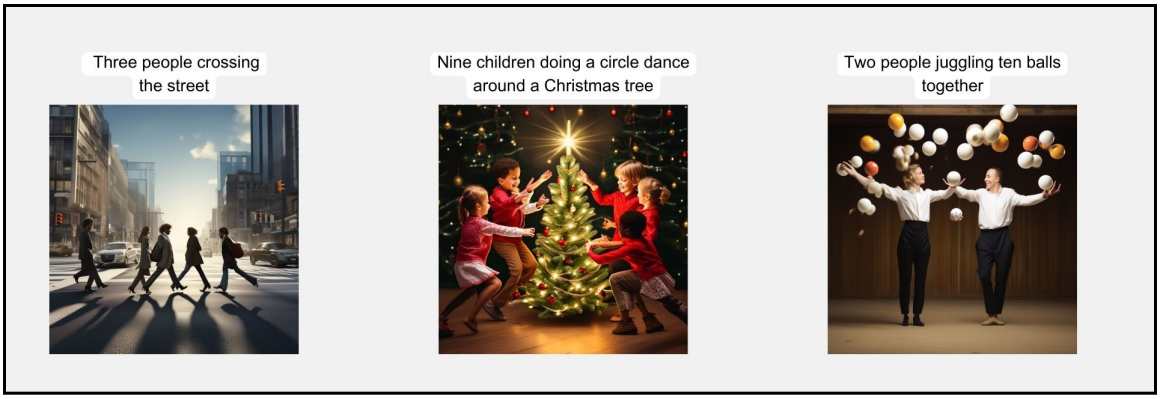


Figure 4.22: The limitations of the Pix-Art-α model for counting tasks at (width, height) = (512,512).


Figure 4.23: Text limitations of the Pix-Art-α model for (width, height) = (1024,1024).


Figure 4.24: Efforts are made to enhance the text limitations of the Pix-Art-α model for (width, height) = (1024,1024).

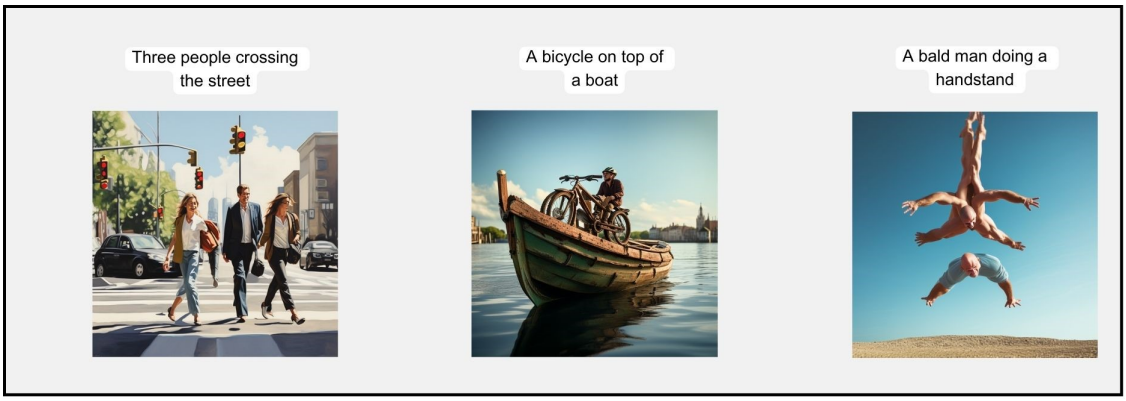


Figure 4.25: The limitations concerning facial characteristics of individuals in the Pix-Art-α model for (width, height) = (1024,1024).

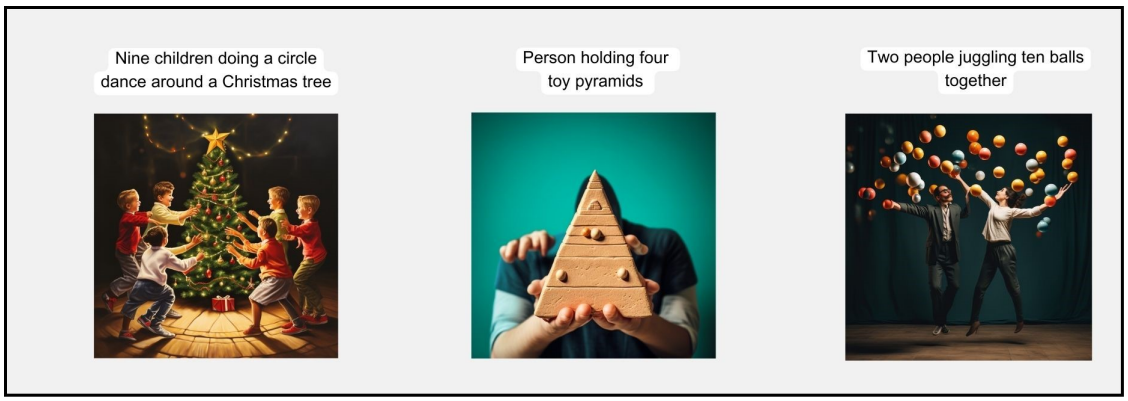


Figure 4.26: The limitations concerning characteristics of people's fingers and hands in the Pix-Art-α model for (width, height) = (1024,1024).

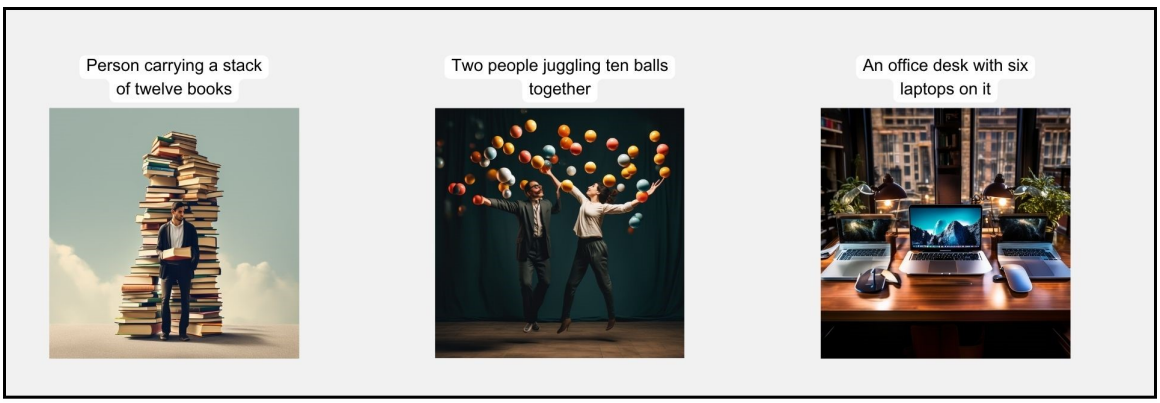


Figure 4.27: The limitations of the Pix-Art-α model for counting tasks at (width, height) = (1024,1024).

# 5. CONCLUSION

We proposed two ways of evaluating text-to-image generative models, human evaluation and clip score metric. Our proposed benchmark allows for testing individual competencies and limitations of the different generative models.

Initially, in the human evaluation, the commercial model DALL-E 2 demonstrates superior performance compared to the other two free models for 512-sized images. DALL-E 2 excels in the tasks of coloring, counting, faces, and text. It effectively understands and reproduces the specified colors and the number of objects. Additionally, it performs very well in the faces task. However, it should be noted that faces are not accurately reproduced when they are not explicitly mentioned in the prompt, and the pupils of the eyes are not always rendered correctly. Furthermore, although its performance in text tasks decreases, it remains higher than that of the other two models. When the image size is increased to 1024, DALL-E 2 continues to outperform the other two models, now excelling in five tasks instead of four as with the 512-sized images. Pix-Art-α is the second-best model based on the human evaluation of 512-sized images. It performs well in two categories: conflicting and positional. This indicates that it understands and accurately reproduces conflicting themes present in the prompt, as well as the spatial arrangement of objects within the image. When the image size is increased to 1024, Pix-Art-α remains the second-best model. However, it excels only in the conflicting task, suggesting that the change in image size negatively affects its performance in the positional task. Stable Diffusion does not perform well in any of the six categories for either 512-sized or 1024-sized images. This indicates poor alignment between the prompts and the generated images.

Additionally, in the CLIP score metric, we observe that the commercial model DALL-E 2 exhibits the best text-image alignment among the three models for the 512-sized images. It achieves the highest score in three tasks: coloring, faces, and text. Furthermore, Pix-Art-α emerges as the second-best model based on the CLIP score, demonstrating good performance in the conflicting and positional tasks. Stable Diffusion excels only in one task, which is counting. Moreover, in the 1024-sized images, DALL-E 2 continues to perform as the best model among the three. It performs well in four tasks: conflicting, counting, positional, and text. Pix-Art-α remains the second-best model based on the CLIP metric, performing well in two tasks: coloring and faces. On the other hand, Stable Diffusion shows poor performance in all tasks.

Understanding the limitations is critical for picking the suitable model for each task and application, advancing the quality of generative models, and aligning their performance with human goals.

# ABBREVIATIONS - ACRONYMS

| | |
|---|---|
| AI | Artificial Intelligence |
| CLIP | Contrastive Language-Image Pretraining |
| D | Discriminator |
| DNN | Deep Neural Network |
| DL | Deep Learning |
| DL2 | DALLE-2 |
| DDPMs | Denoising Diffusion Probabilistic Models |
| FID | Fréchet Inception Distance |
| FM | Flow Matching |
| GPT | Generative Pre-trained Transformer |
| GAI | Generative Artificial Intelligence |
| GANs | Generative Adversarial Networks |
| G | Generator |
| IS | Inception Score |
| LLM | Large Language Model |
| ML | Machine Learning |
| NL | Natural Language |
| NN | Neural Network |
| PA | Pix-Art-α |
| PL | Programming Language |
| prt | prompt |
| SD | Stable Diffusion |
| 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 |
| Std | Standard Deviation |

# REFERENCES


[1] Openai api documentation. URL: https://platform.openai.com/docs/api-reference.

[2] 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. Music lm: Generating music from text. URL: https://arxiv.org/abs/2301.11325, 2023.

[3] AUTOMATIC1111. stable-diffusion-webui, 2023. URL: https://github.com/AUTOMATIC1111/stable-diffusion-webui.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. URL: https://arxiv.org/abs/1607.06450, 2016.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. URL: https://arxiv.org/abs/1409.0473, 2016.

[6] 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. URL: https://arxiv.org/abs/2211.01324, 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. URL: https://www.mdpi.com/1999-5903/15/8/260.

[8] 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. URL: https://www.mdpi.com/1999-5903/15/8/260.

[9] Shane Barratt and Rishi Sharma. A note on the inception score. URL: https://arxiv.org/abs/1801.01973, 2018.

[10] Michael Cahyadi, Muhammad Rafi, William Shan, Jurike Moniaga, and Henry Lucky. Accuracy and fidelity comparison of luna and dall-e 2 diffusion-based image generation systems. URL: https://arxiv.org/abs/2301.01914, 2023.

[11] 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. URL: https://arxiv.org/abs/2303.04226, 2023.

[12] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-$\alpha$: Fast training of diffusion transformer for photorealistic text-to-image synthesis. URL: https://arxiv.org/abs/2310.00426, 2023.

[13] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. URL: https://arxiv.org/abs/1904.10509, 2019.

[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. URL: https://arxiv.org/abs/2108.06209, 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] CompVisi. Stable Diffusion v1 Model Card. URL: https://github.com/CompVis/stable-diffusion/blob/main/Stable_Diffusion_v1_Model_Card.md#stable-diffusion-v1-model-card, 2022.

[17] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. URL: https://arxiv.org/abs/2005.00341, 2020.

[18] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. URL: https://arxiv.org/abs/2210.13438, 2022.

[19] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. URL: https://arxiv.org/abs/2302.03011, 2023.

[20] B. Brown et al. Language models are few-shot learners. URL: https://arxiv.org/abs/2005.14165, 2020.

[21] Bommasani et al. On the opportunities and risks of foundation models. URL: https://arxiv.org/abs/2108.07258, 2022.

[22] Chen et al. Evaluating large language models trained on code. URL: https://arxiv.org/abs/2107.03374, 2021.

[23] Chitwan Saharia et al. Photorealistic text-to-image diffusion models with deep language understanding. URL: https://arxiv.org/abs/2205.11487, 2022.

[24] Chowdhery et al. Palm: Scaling language modeling with pathways. URL: https://arxiv.org/abs/2204.02311, 2022.

[25] Srivastava et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. URL: https://arxiv.org/abs/2206.04615, 2023.

[26] 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. URL: https://arxiv.org/abs/2002.08155, 2020.

[27] Stefan Feuerriegel, Jochen Hartmann, Christian Janiesch, and Patrick Zschech. Generative ai. *Business, Information Systems Engineering*, 66, September 2024. URL: http://dx.doi.org/10.1007/s12599-023-00834-7.

[28] 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. URL: https://arxiv.org/abs/2203.13131, 2022.

[29] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. URL: https://arxiv.org/abs/1701.00160, 2017.

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

[31] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. URL: https://arxiv.org/abs/1406.2661, 2014.

[32] Roberto Gozalo-Brizuela and Eduardo C. Garrido-Merchán. A survey of generative ai applications. URL: https://arxiv.org/abs/2306.02781, 2023.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. URL: https://arxiv.org/abs/1512.03385, 2015.

[34] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. URL: https://arxiv.org/abs/1706.08500, 2018.

[35] 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. URL: https://arxiv.org/abs/2210.02303, 2022.

[36] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. URL: https://arxiv.org/abs/2006.11239, 2020.

[37] 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. URL: https://arxiv.org/abs/2208.12415, 2022.

[38] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. URL: https://arxiv.org/abs/1812.04948, 2019.

[39] 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. URL: http://dx.doi.org/10.1038/s41524-020-0276-y.

[40] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. URL: http://dx.doi.org/10.1561/2200000056.

[41] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. URL: https://arxiv.org/abs/1312.6114, 2022.

[42] 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. URL: https://arxiv.org/abs/2306.15687, 2023.

[43] 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. URL: https://arxiv.org/abs/2211.10440, 2023.

[44] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. URL: https://arxiv.org/abs/2210.02747, 2023.

[45] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. URL: https://arxiv.org/abs/2304.08485, 2023.

[46] 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. URL: https://www.sciencedirect.com/science/article/pii/S2352847823000771.

[47] Zhihan Lv. Generative artificial intelligence in the metaverse era. *Cognitive Robotics*, 3:208–217, 2023. URL: https://www.sciencedirect.com/science/article/pii/S2667241323000198.

[48] 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. URL: https://arxiv.org/abs/2003.08934, 2020.

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

[50] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: https://probml.github.io/pml-book/book1.html.

[51] 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. URL: https://arxiv.org/abs/2112.10741, 2022.

[52] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. URL: https://arxiv.org/abs/2212.08751, 2022.

[53] 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. URL: https://arxiv.org/abs/2203.13474, 2023.

[54] openai. CLIP. URL: https://github.com/openai/CLIP.

[55] OpenAI. Dall-e 3 system card. URL: https://openai.com/research/dall-e-3-system-card, 2023.

[56] OpenAI and : Achiam et al. Gpt-4 technical report. URL: https://arxiv.org/abs/2303.08774, 2023.

[57] William Peebles and Saining Xie. Scalable diffusion models with transformers. URL: https://arxiv.org/abs/2212.09748, 2023.

[58] Vitali Petsiuk, Alexander E. Siemenn, Saisamrit Surbehera, Zad Chin, Keith Tyser, Gregory Hunter, Arvind Raghavan, Yann Hicke, Bryan A. Plummer, Ori Kerret, Tonio Buonassisi, Kate Saenko, Armando Solar-Lezama, and Iddo Drori. Human evaluation of text-to-image models on a multi-task benchmark. URL: https://arxiv.org/abs/2211.12112, 2022.

[59] PixArt. Pixart-alpha 1024px. URL: https://huggingface.co/spaces/PixArt-alpha/PixArt-alpha.

[60] PixArt. Pixart-$\alpha$ model card. URL: https://huggingface.co/PixArt-alpha/PixArt-XL-2-1024-MS.

[61] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. URL: https://arxiv.org/abs/2209.14988, 2022.

[62] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. URL: https://arxiv.org/abs/2103.00020, 2021.

[63] 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. URL: https://arxiv.org/abs/1910.10683, 2023.

[64] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. URL: https://arxiv.org/abs/2204.06125, 2022.

[65] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. URL: https://arxiv.org/abs/2102.12092, 2021.

[66] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. URL: https://arxiv.org/abs/1906.00446, 2019.

[67] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. URL: https://arxiv.org/abs/2112.10752, 2022.

[68] Robin Rombach and Patrick Esser. Huffing face, stable-diffusion-v1-5 model card, 2023. URL: https://huggingface.co/runwayml/stable-diffusion-v1-5.

[69] 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. URL: https://arxiv.org/abs/2205.11487, 2022.

[70] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. URL: https://arxiv.org/abs/1606.03498, 2016.

[71] Haroon Sheikh, Corien Prins, and Erik Schrijvers. *Mission AI: The New System Technology*. Springer International Publishing, 2023. URL: https://doi.org/10.1007/978-3-031-21448-6_2.

[72] 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. URL: https://arxiv.org/abs/2209.14792, 2022.

[73] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. URL: https://arxiv.org/abs/1503.03585, 2015.

[74] C. O. S. Sorzano, J. Vargas, and A. Pascual Montano. A survey of dimensionality reduction techniques. URL: https://arxiv.org/abs/1403.2877, 2014.

[75] The European Commission. *A definition of AI: Main capabilities and scientific disciplines*. The European Commission, 2019. URL: https://digital-strategy.ec.europa.eu/en/library/definition-artificial-intelligence-main-capabilities-and-scientific-disciplines.

[76] 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. URL: https://arxiv.org/abs/2302.13971, 2023.

[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. URL: https://arxiv.org/abs/1706.03762, 2023.

[78] 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. URL: https://arxiv.org/abs/2210.02399, 2022.

[79] 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. URL: https://arxiv.org/abs/2301.02111, 2023.

[80] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. URL: https://arxiv.org/abs/2312.02201, 2023.

[81] 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. URL: https://arxiv.org/abs/2109.00859, 2021.

[82] BigScience Workshop and : Le Scao et al. Bloom: A 176b-parameter open-access multilingual language model. URL: https://arxiv.org/abs/2211.05100, 2023.

[83] 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. URL: https://arxiv.org/abs/2209.00796, 2024.

[84] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. URL: https://arxiv.org/abs/2107.03312, 2021.

[85] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. URL: https://arxiv.org/abs/2104.03670, 2021.