UNIVERSITY OF PADOVA

Department of General Psychology

Master Degree in Cognitive Neuroscience and Clinical Neuropsychology

Final dissertation

Evaluation of basic mathematical abilities of neural networks

Supervisor
Professor Alberto Testolin

Candidate: Kuinan Hou
Student ID number: 2040699

Academic Year 2022/2023

Abstract

In human cognition, basic numerical skills, such as number sense and elementary calculation, are typically well-developed when advanced mathematical abilities reach a certain level. In this thesis, we investigate whether state-of-the-art artificial neural network models exhibit a similar trend. Indeed, much research has pointed out that large-scale language models (such as ChatGPT) possess exceptional high-level mathematical abilities, but their elementary numeracy skills have often been overlooked. This dissertation focuses on the foundational mathematical abilities of GPT-3.5 (from which ChatGPT was developed), its newest version GPT-4 and six other multi-modal deep learning models. Taking into account the unique characteristics of different neural network models, standardized tests and self-developed tasks were employed to explore the mathematical abilities of these eight models. The findings indicate that GPT-3.5 and GPT-4 are indeed able to exhibit complex mathematical competencies, though basic numeracy skills are not always fully developed (especially in GPT-3.5). In contrast, the six multi-modal models still need to make progress in improving their numerosity perception and number sense to unlock more advanced mathematical abilities.

Contents

Introduction

Mathematics is often referred to as the ”language of science” and serves as a fundamental tool in many scientific disciplines (Knuth, 1974; Riley et al., 1999). It provides a universal language and a set of tools that allow scientists to explain complex phenomena and establish the validity of their theories and experiments. Given the increasingly broad and important role that AI plays in various aspects of scientific research and the importance of modern neural network models in studying human cognition, Artificial Intelligence (AI) competence should be thoroughly considered.

1.1 General Numerical and Mathematical Abilities

1.1.1 Numerosity Perception

Numerosity perception refers to an organism’s ability to perceive and quantify the number of objects in a set or group (Gebuis et al., 2016). This cognitive function is crucial for various daily activities, including decision-making and problem-solving. Research has shown that numerosity perception occurs in two ranges: subitizing and estimation. Subitizing refers to the rapid and accurate perception of small quantities (usually up to four) (Kaufman et al., 1949), while estimation refers to the ability to judge larger quantities, although less accurately (Trick & Pylyshyn, 1994). The same as other perceptions, after perceiving them correctly, cognition starts to function. The cognitive process that includes perception and the subsequent process of numerosity can be defined as the number sense. Number sense is a broad concept that many scholars define differently (Gersten & Chard, 1999). Berch (2005), after reviewing many academic papers, define the number sense as the ability to grasp everything from comprehending the essence of numbers to devising methods for solving intricate mathematical problems; from carrying out straightforward magnitude comparisons to creating protocols for performing numerical operations; and from identifying obvious numerical mistakes to utilizing quantitative approaches for exchanging processing, and deciphering information. Although closely related, these processes are believed to be mediated by separate neural mechanisms (Piazza et al., 2010). In the brain, the parietal cortex has been identified as a key region for numerosity perception, specifically the intraparietal sulcus (IPS) (Nieder, 2016). Neuroimaging studies have found that IPS is activated during tasks that require numerical judgments, further supporting its role in numerosity perception (Dehaene et al., 2003). However, the mechanism of numerosity perception is still a topic of active research. There are several theories about how our brains perceive numerosity, including the Approximate Number System (ANS) theory, which suggests that we possess an inherent sense of number that allows us to estimate the numerosity of sets (Dehaene, 2011).

From a developmental perspective, the number sense emerges in early childhood and continues to refine throughout schooling. Even infants demonstrate a rudimentary number sense when they distinguish between different quantities (Starkey & Cooper Jr, 1980). This inherent understanding of quantities is the starting point for developing the sense of numbers. As children get older and gain education, they begin to understand more complex numerical concepts such as place values, fractions, and arithmetic operations, further enhancing their sense of numbers (Jordan et al., 2007). Some scholars would think that numerosity perception and number sense are innate since these abilities can also be found in nonhuman animals (Agrillo & Bisazza, 2018; Brannon, 2005). However, artificial neural networks can also possess the ability to discriminate the numerosity with visual inputs (Testolin et al., 2020), which is fully learned. Simulation of visual perception of numerosity using an artificial neural network returns similar behavioral patterns shown by animals and humans(Stoianov & Zorzi, 2012). In fact, during the past few years, many neural networks have shown incredibly good high-level mathematical abilities, such as automated theorem proving (Wang & Deng, 2020); solving ordinary differential equations (Lample & Charton, 2019) etc. These models make us wonder how neural networks develop number sense and do neural networks have a number sense similar to ours.

1.1.2 Multimodal Numerosity Perception

Numerosity, or the perception of quantity, is a fundamental cognitive ability that transcends species and sensory modalities (Starr et al., 2013). This ability to process and perceive numerical information from various sensory inputs is referred to as multimodal numerosity perception. For example, humans can estimate the number of objects in a set using visual and auditory stimuli, such as seeing a group of birds on a wire and simultaneously hearing their individual chirps. It is suggested that this ability is also facilitated by ANS (Burr & Ross, 2008).

Research also indicates the possibility of cross-modal perception of numerosity, where sensory information from one modality can influence the perception of numerosity in another. For instance, the sound of several objects falling might influence our visual estimate of the number of objects present (Nieder, 2016).

The multimodal perception of numerosity plays an essential role in our daily lives, underpinning a variety of tasks, from basic arithmetic to more complex problem-solving. Understanding the mechanisms behind this cognitive process can also help researchers design more effective learning tools and strategies, potentially improving educational outcomes and facilitating cognitive development.

1.1.3 Mathematical Abilities and Number Sense

Recent research suggested that mathematical abilities might be closely intertwined with numerosity perception. Some have argued that the ability to understand and manipulate numbers, or numerical cognition, forms the basis for more complex mathematical skills (Feigenson et al., 2004). As shown in Figure 1.1, the higher the number acuity, the higher the score on math tests. This sense of numbers is believed to be fundamental to the development of mathematical abilities. Studies have shown a correlation between the acuity of an individual's ANS and their performance on standardized mathematics tests (Halberda et al., 2008). It is also crucial because of its strong connection to broader mathematics achievement. Students with a well-developed number sense are more likely to succeed in mathematics, while a lack of number sense can lead to difficulties in mathematics (Jordan et al., 2010).

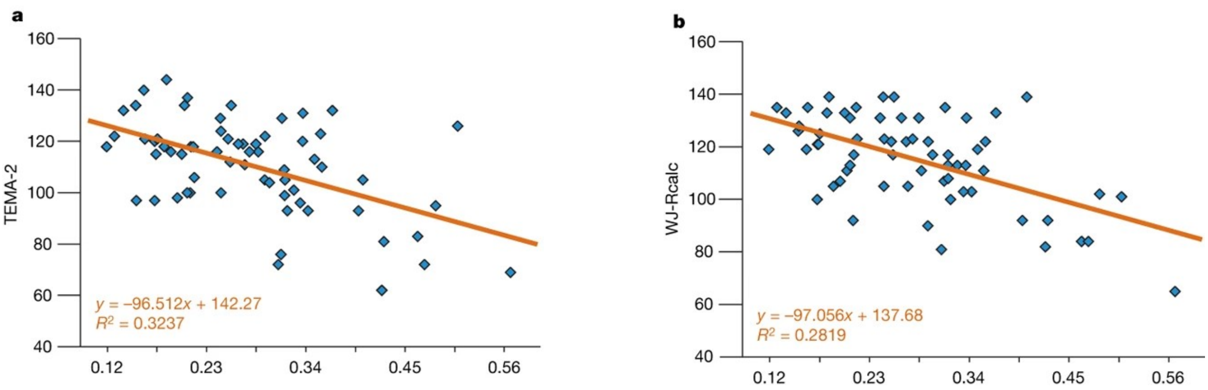

Figure 1.1: linear regression analysis for two different symbolic math achievement tests, namely TEMA-2 and WJ-Rcalc. In both cases, the horizontal axis is the Weber fraction; higher scores signify better performance. The analysis looks at the standard score of each subject on these tests and relates it to the acuity of the Approximate Number System (ANS), denoted by 'w'. For the ANS Weber fraction, lower scores denote better performance. TEMA-2: A test of early mathematical ability, second edition. WJ-Rcalc: Woodcock–Johnson revised calculation subtest. From "Individual differences in non-verbal number acuity correlate with math achievement," by Halberda, Mazzocco, & Feigenson, 2008

It involves mapping symbolic numbers to the corresponding numerical magnitudes, which is a crucial skill in arithmetic (Nieder, 2016). Furthermore, numerosity perception plays a significant role in early mathematical learning. It is suggested that an intuitive understanding of the numbers of six-month-old babies can predict their later mathematical achievement (Starr et al., 2013) (see Figure 1.2). Therefore, early interventions aimed at improving number perception might have a positive effect on a child's subsequent math performance.

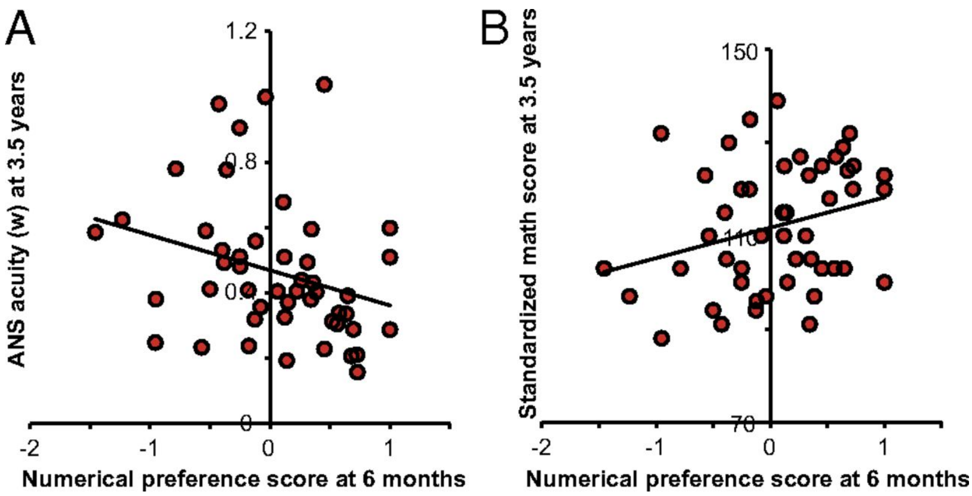

Figure 1.2: Comparison of how the numerical preference of infants is correlated with the ANS acuity and math score in the future. From "Number sense in infancy predicts mathematical abilities in childhood," by Starr, Libertus, & Brannon, 2013

In this paper, the general mathematical ability of neural networks is investigated. The nature of those tasks could be pure numerosity perception number sense, or a mix of both.

1.1.4 Geometrical Ability

Geometric ability, a key facet of spatial cognition, refers to the ability to understand, interpret, and manipulate geometric shapes and principles. It involves the recognition and discrimination of shapes, understanding the properties and relationships of shapes and angles, mental rotation of objects, and perception of spatial orientation and direction (Linn & Petersen, 1985).

This cognitive ability plays a fundamental role in many everyday tasks and academic disciplines. For instance, it is crucial for navigation, understanding maps, interpreting diagrams, and appreciating art. In academic realms, geometrical ability is vital for fields such as mathematics, physics, engineering, and architecture. Furthermore, it is necessary for spatial reasoning tasks in various professions, such as surgeons, pilots, and artists (Uttal et al., 2013).

Research suggests that geometric ability can be enhanced with targeted training. For example, engaging in spatially demanding activities such as puzzles, video games, or specific mental rotation exercises can improve geometrical skills (Newcombe et al., 2015).

An understanding of geometrical ability and its development can also inform educational practices, potentially leading to improved outcomes in STEM fields, science, technology, engineering, and mathematics, which are highly based on spatial reasoning.

1.2 Neural Networks and Mathematical Abilities

1.2.1 Neural Networks

Neural networks are a subset of machine learning methods inspired by the structure and functionality of the human brain, designed to recognize patterns and interpret complex data. These artificial networks are interconnected layers of nodes or "neurons," which work together to solve specific problems (Goodfellow et al., 2016). A typical neural network consists of an input layer, one or more hidden layers, and an output layer. Neural networks with more than one hidden layer are considered deep neural networks, and the process of training them is deep learning. Each node in a layer is connected to nodes in the next layer through ”synapses,” which are associated with weights that determine the influence of one node on another. When presented with data, the network learns by adjusting these weights based on the difference between its predicted and actual outputs, a process known as training (Schmidhuber, 2015).

Neural networks are used in a variety of applications, from speech and image recognition to natural language processing and autonomous vehicles. In this thesis, we will focus on transformers, diffusion models, and visual question-answering models. All of these models consist of a basic feedforward neural network.

1.2.2 Feedforward Neural Networks (FNN)

Feedforward Neural Networks (FNN) is the simplest form of Artificial Neural Networks (ANNs), with data moving in only one direction (see Figure 1.3, from the input to the output. The main application of FNNs is in pattern recognition and classification. An example of this can be seen in the task of classifying handwritten digits.

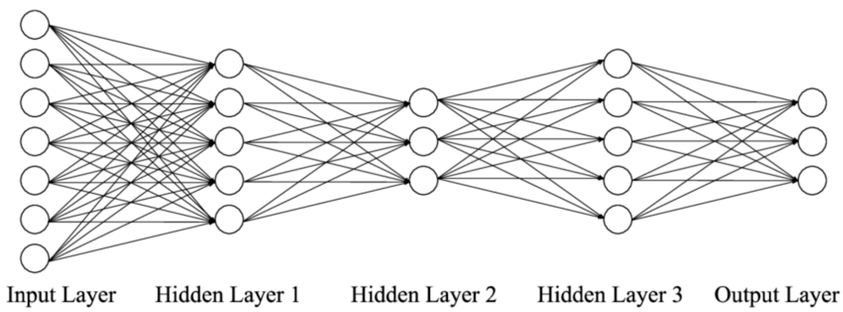

Figure 1.3: An illustration of FNN

1.2.3 Transformers

As for Transformers (Vaswani et al., 2017), in particular, both the human brain and Transformers exhibit abilities that are crucial in processing mathematical problems, although through fundamentally different mechanisms. They are proficient at handling sequences, a skill that is critical when breaking down a mathematical problem into simpler solvable steps. This sequential processing is a hallmark of both human problem-solving and the operation of

Transformer models. Furthermore, both systems effectively manage symbolic representations. In the realm of mathematics, symbols denote numbers, operations, and relationships. Humans learn to interpret and manipulate these symbols to resolve problems, a process mirrored by Transformers, which treat various forms of input as symbolic tokens and learn to discern patterns and relationships among them. Pattern recognition is another shared competency. Humans are capable of discerning patterns in mathematical problems, which enables the application of known strategies or heuristics. Similarly, Transformer models learn to recognize data patterns during their training, which they then use to generate predictions or solve problems. Lastly, both humans and Transformers demonstrate the ability to generalize from learned experiences. Once a human grasps a mathematical concept, they can typically apply it to new, yet similar, problems. Similarly, Transformers are designed to extrapolate from their training data to tackle new, unseen data. Despite these similarities, it is important to remember that Transformers, while capable of simulating some aspects of human cognition, do not truly ”understand” math problems or experience consciousness in the way humans do. Instead, they employ statistical patterns learned during their training to produce outputs for specific inputs.

Prompting

Nowadays LLMs, such as GPT models, are developed on the basis of transformers. For those LLMs, a ”prompt” refers to the initial input text that is fed into the model to generate a specific output text. The prompt serves as the basis or cue that guides the model in generating a response. The model’s response is generated by predicting the next set of words based on the given prompt, guided by the weights and patterns the model learned during its training phase(Radford et al., 2019).

Research has shown that rather than individually fine-tuning a language model for each new task, it is possible to guide the model using a handful of input-output examples within the prompt. Simply adding a few examples in the prompt, the performance of GPT-3 has been improved(Brown et al., 2020).

Inspired by the work of Brown et al, many other prompting techniques have been invented to improve LLMs’ performance. Here are 2 recent findings.

Chain-of-Thought Prompt (CoT) is a method for improving the ability of large language models to perform complex reasoning by providing examples of intermediate reasoning steps. which can significantly improve the performance on a variety of arithmetic, commonsense, and symbolic reasoning tasks(Wei et al., 2022).

Progressive-Hint Prompting (PHP) allows for automated, multi-step interactions between users and LLMs by utilizing past responses as hints to incrementally steer toward the right answers. PHP is compatible with other prompting techniques like CoT, making it straightforward to integrate with state-of-the-art methods for enhanced performance(Zheng et al., 2023).

1.2.4 Diffusion Models

Diffusion models, a class of stochastic processes, have played an increasingly critical role in the field of artificial intelligence, particularly in the realm of generative modeling(Sohl-Dickstein et al., 2015). The strength of these models lies in their ability to model the data distribution via a stochastic process known as diffusion, enabling them to generate high-quality samples that closely mimic the underlying distribution of the training data. It is typically operated through a Markov chain Monte Carlo (MCMC) procedure, which iteratively refines a random initial point until it converges to a sample from the target distribution(Du et al., 2023). The process begins with an observable data point and adds Gaussian noise at each step, effectively 'blurring' the data point over time. This noisy path is then traversed in reverse, starting from a simple distribution and gradually denoising it to recover a sample from the complex data distribution.

This structure gives diffusion models several significant advantages. The iterative refinement process allows the model to gradually shape the sample, correcting errors and improving quality over time. This is a contrast to many generative models that produce a sample in a single shot and have no opportunity to correct errors once they are made (Sohl-Dickstein et al., 2015).

In terms of applications, diffusion models have been employed across a wide range of domains, including image synthesis (see Figure 1.4, text generation, and, more recently, molecular dynamics and financial modeling. For instance, in image synthesis, they have been used to generate high-quality, photorealistic images that are almost indistinguishable from real images(Song

Figure 1.4: Examples from Stable diffusion. On the left, the prompt is ”dog with a hat”, on the right, the prompt is ”astronaut riding a horse”

et al., 2020).

A prominent extension of diffusion models is conditioned diffusion models, which condition the diffusion process on auxiliary inputs. These models have demonstrated impressive performance on a range of tasks, from image synthesis to text generation, among others(Dhariwal & Nichol, 2021). The ability to condition the generative process based on given inputs allows these models to generate contextually relevant output, enabling a more targeted generative process.

The power of diffusion models and conditioned diffusion models in complex generative tasks cannot be understated. For example, in tasks such as text generation, the output space is typically multimodal, with many plausible outputs. Diffusion models excel in these tasks as they can naturally model these multimodal distributions(Dhariwal & Nichol, 2021). Their ability to model intricate data distributions and generate high-quality output makes them a valuable tool in the ever-evolving landscape of artificial intelligence.

1.2.5 Visual Question Answering Models

Visual Question Answering (VQA) models are a type of AI model designed to understand and answer questions about visual content. At their core, these models must solve two fundamental problems: understanding visual data (usually images or videos) and understanding the natural language question posed about the data. To do this, most VQA models consist of three main components: an image feature extractor, a question understanding module, and an answer

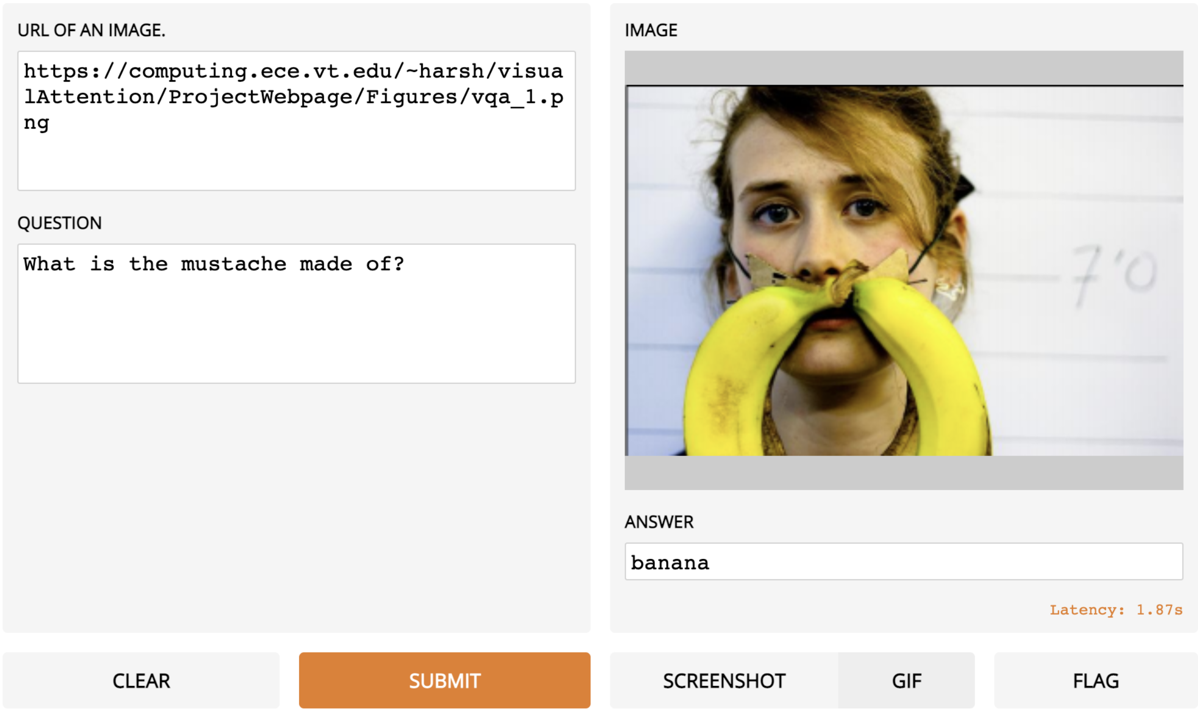

Figure 1.5: An example of what VAQ can do.

prediction module (Antol et al., 2015).

The image feature extractor usually uses Convolutional Neural Networks (CNNs) to capture the semantic and spatial features of the image. The question understanding module, on the other hand, typically employs recurring neural networks (RNNs) or, more recently, transformer-based models to understand the semantics and syntactic structures of the question. The features extracted from both modules are then combined, usually by a multimodal fusion technique, to understand the correlation between the visual content and the question. The result is used by the answer prediction module to generate a suitable response (Zhou et al., 2020).

The applications of VQA models are broad and impactful. They play an essential role in human-computer interaction, allowing users to interact with computers using natural language about visual content. Figure 1.5 shows the input and output of a VQA model. In addition, they can be used in accessibility technologies for visually impaired individuals, facilitating their understanding of visual content through natural language interfaces. In the realm of autonomous vehicles and drones, VQA models can provide richer environmental understanding by allowing systems to ask and answer questions about the visual world (Ganesh et al., 2019; Sarkar et al., 2023).

Despite their success, VQA models still face significant challenges. These include understanding ambiguous questions, dealing with multi-modal data fusion, and handling questions that require commonsense or external knowledge. However, advances in deep learning and the development of large-scale VQA data sets promise a steady stream of innovations in the field.

1.2.6 Parallel Distributed Processing

Neural networks are good at simulating the cognitive process of the human brain for various reasons. Parallel Distributed Processing (PDP), also known as connectionism, is a computational approach to understanding cognitive processes such as perception, memory, language, and thought (McClelland et al., 1986). PDP models are based on the idea that cognitive processes result from the simultaneous operation of a large number of simple processing units that are interconnected in complex networks. Each unit in a PDP model represents a neuron-like element that takes input, processes it, and sends output to other units (Rumelhart et al., 1986). PDP models are a type of artificial neural network. They are composed of layers of nodes or "neurons," each layer connected to the next. Information flows through these connections, and the strength of each connection (or "weight") determines how much influence one node has on another. PDP models learn by adjusting these weights based on the difference between the output produced and the desired output. This learning algorithm, known as backpropagation, is used in many types of neural networks.

Several key features of PDP models also apply to artificial neural networks in general:

Distributed Representation: Information is not stored in a single unit or location but is distributed across many units. This feature gives neural networks their robustness and the ability to generalize from learned experiences to new ones.

Parallel Processing: Multiple computations are carried out simultaneously, which is believed to mimic parallel processing of the human brain and leads to efficient information processing.

Adaptive Learning: Networks can learn from experiences and adapt their responses accordingly. This adaptability comes from the ability to adjust the strengths of the connections between units based on experience.

In summary, PDP models provide an approach to understanding cognitive processes based on principles that are also fundamental to neural networks, emphasizing distributed representation, parallel processing, and adaptive learning.

1.2.7 Geometrical Ability of Neural Networks

Deep neural networks, particularly Convolutional Neural Networks (CNNs), excel in processing geometric visual data. CNNs are designed with layers of artificial neurons or ”nodes” that apply a series of filters to the input. These filters allow the network to detect edges, curves, and other simple geometric features in the early layers. As the information flows further into the network, these features are combined to recognize more complex geometric structures. This hierarchical layered processing is believed to be similar to how our brains process visual information (Kriegeskorte, 2015).

Another aspect of neural networks' geometrical ability is their capacity for transformation invariance, meaning that they can recognize a shape or object regardless of its position, size, or orientation in the visual field. This ability is essential for many machine vision tasks, including object detection, facial recognition, and handwriting recognition.

Moreover, recent advances in neural networks incorporate more sophisticated geometric understanding. For example, Graph Neural Networks (GNNs) and Geometric Deep Learning approaches have been designed to handle data structured as graphs, manifolds, or other complex geometrical constructs, which are common in fields such as social network analysis, molecular chemistry, and 3D computer vision (Bronstein et al., 2021).

1.2.8 Challenges for Neural Networks

Inductive bias in machine learning refers to the set of assumptions that a learner uses to predict the output given inputs that it has not encountered before. It is the bias or set of preconceptions that a machine learning algorithm has, which guides the algorithm towards a particular solution when faced with uncertain data. Inductive bias is essential to make predictions in situations where the possible hypotheses exceed the available data.

There are two main types of inductive bias:

Preference Bias: This type of bias reflects a preference for some hypotheses over others, often ordering them according to their complexity. For example, in decision tree learning, the preferred hypothesis is usually the smallest tree that correctly classifies the training examples.

Restriction Bias: This type of bias restricts the space of the hypothesis by defining a set of possible hypotheses that the learner may consider. For example, in linear regression, the restriction bias is the assumption that the target function is a linear combination of the input features.

Without some form of inductive bias, a learning algorithm would not be able to generalize from the training data to unseen data, and machine learning would be virtually impossible."Neural networks, by virtue of their architecture, demonstrate a form of inductive bias that guides the learning process and the way these systems generalize from seen to unseen data (Goodfellow et al., 2016). Although these systems can effectively capture patterns in the data, their performance in mathematical tasks has been questioned. In fact, mathematical reasoning often requires abstract thinking and symbolic reasoning that go beyond pattern recognition (Marcus, 2003). Such tasks might involve extrapolating from limited examples to an entire class of problems, which is different from the kind of generalization neural networks typically perform (LeCun et al., 2015).For instance, a network trained on a large amount of data involving addition problems might fail to apply this understanding to multiplication problems if it hasn't been exposed to enough multiplication data (Lake et al., 2017). This is a clear manifestation of the inductive bias of neural networks that limits their ability to generalize beyond specific types of problems and mathematical operations encountered during training. Moreover, due to their inductive bias, neural networks usually interpolate between examples in the training set rather than extrapolating to unseen situations. This characteristic hampers their performance on tasks such as mathematical theorem proving, which requires extrapolation from known rules to new situations. Thus, the inductive bias inherent in neural networks can limit their mathematical abilities, especially when it comes to generalizing beyond the specific types of problem and mathematical operations seen during training (Goodfellow et al., 2016; LeCun et al., 2015)." Another challenge that is commonly encountered by Transformers or any models that apply the recursive structure is exposure bias, also known as teacher forcing in the context of machine learning, particularly in sequence-to-sequence models, referring to a discrepancy between the data distribution the model is trained on and the distribution it encounters when it generates sequences during testing. During training, the model is usually fed the ground truth of the previous time step as input. This means that at each step the model bases its predictions on the actual historical sequence. This is known as teacher forcing. However, when the model is used for prediction or inference, it does not have access to the ground truth. Instead, it takes its own output from the previous time step as input for the current step. This difference in data distribution between the training and inference stages is known as exposure bias. Exposure bias can lead to the model accumulating errors over time when generating sequences during inference, which can significantly degrade the quality of the output (Schmidt et al., 2019).

Inaccurate number sense From the perspective of performance, although recent neural networks perform well in high-level mathematical problems when it comes to the sense of basic numbers (Lample & Charton, 2019; Shakarian et al., 2023), neural networks fail to do many simple and trivial tasks (Frieder et al., 2023). This thesis will test the number sense of neural networks in a detailed way.

Materials and Methods

All numbers that are generated randomly were generated with the Python Random library whose seed was set to 13. Inputs are carefully tokenized to get a better result, which includes adding space between numbers and operators; and changing *:* to ÷ when comes to division.

An example is provided in the prompt for those poorly answered questions and some detailed and easier tasks are instructed for those failed tasks. For the tasks that are well performed, more challenging tasks are instructed. These tests can be found in the Customized Data Sets section of this chapter.

2.1 Number Screening

The Number Screening consists of 4 sections, which is a general screening for basic numerical abilities.

Counting list This section tests the ability to count numbers and items voluntarily.

Couting numbers

"count from *number_a* to *number_b*"

"count from *number_a* to *number_b by number_c*"

Comprehension of Numbers This section tests the ability to understand and manipulate numbers.

Parity Judgement

"is *number_a* odd or even"

Number Comparison Arabic

”Which is bigger, *number_a* or *number_b?”*

Number Comparison Word

”Which is bigger, *number_a* or *number_b*? ”

NB: numbers are in written format: three instead of 3

Numerical Transcoding This section tests the ability to transcode mental numbers into different explicit forms and the other way around.

Arabic numerals → Token

”We have three types of colored tokens. The big green one corresponds to 100, the intermediate orange one corresponds to 10 and the small black one corresponds to 1. how to represent *number_a* with these tokens?”

Token → Arabic numerals

”We have three types of colored tokens. The big green one corresponds to 100, the intermediate orange one corresponds to 10 and the small black one corresponds to 1. which number does it represent if we have 0 green token(s), 0 orange token(s), and 3 black token(s)?”

Arabic numerals → Words

”How to read *number_a* in English?”

Words → Arabic numerals

”how to represent *number_a* in Arabic numerals?

NB: numbers are in written format: three instead of 3”

Arithmetic Facts This section tests the facts and rules of 4 basic operations and the wider mathematical application (i.e. approximation) with 4 basic operations. Facts refer to the calculation and rules refer to the rules (e.g. 0 times anything equals 0; inverting the order of 2 numbers in the multiplication does not change the result, etc.).

”calculate: *number_a* ÷ *number_b*”

”among the following numbers: *number_a*, *number_a*, *number_b*, *number_c*, *number_d*

which is the closest approximation of the operation: *number_x* + *number_y*”

2.2 Customized number screening tasks

Some tasks are trivial for human beings, but tricky for language models (see Figure 3.1). Thus, to better understand the mathematical abilities of language models and why these models fail in some tasks, the original tasks in the Number Screening are simplified or simple prompting techniques are used to help language models. On the other hand, some tasks in the Number Screening may be too easy for language models, as a result, 2 approaches are adopted to challenge the models. One is to increase the numerosity being tested, another is adding some tests that are not in the Number Screening.

By applying these approaches, we wish to understand 2 language models' upper bound and their shortage of basic mathematical abilities.

2.2.1 Disclosure of complex tasks

Promting with example It is possible that the model already can solve a question, but due to the lack of exposure to this type of question, the model cannot perform well. Thus, giving an example in the prompting that shows the logic and reasoning process of solving this type of question should improve the performance. As in this thesis, the transcoding between tokens to Arabic numerals adopts this technique. Below are examples of how this technique is implemented in the prompts.

Without Example

”We have three types of colored tokens. The big green one corresponds to 100, the intermediate orange one corresponds to 10 and the small black one corresponds to 1. which number does it represent if we have 0 green token(s), 0 orange token(s), 3 black token(s).”

With Example

”We have three types of colored tokens. The big green one corresponds to 100, the intermediate orange one corresponds to 10 and the small black one corresponds to 1. For example, 3 green token(s), 4 orange token(s) and 1 black token represent 341 because 300 + 40 + 1 = 341.

which number does it represent if we have 0 green token(s), 0 orange token(s), 3 black token(s)."

Deconstructed Tasks

Approximation without calculation The original approximation task requires more than one facet of basic mathematical ability. The first is a rough estimate of the result of the calculation, followed by a selection of the four numbers that most closely resemble the estimate. This involves the process of estimating twice. To get a clear picture of how the language models performed in this task, we break the original one-piece estimation process into two separate estimation processes and test the pure estimation process, that is, to estimate the distance of 2 numbers on the number line. An example is provided here.

"among the following numbers: *number_a*, *number_a*, *number_b*, *number_c*, *number_d*, which is the closest to *number_x*"

Feature removal in transcoding between Arabic numerals and tokens There are 2 features, the size and color of a token, that can be used to transcode between Arabic numerals and tokens in the original task. For humans, this could be a boon for the brain but for language models, this may cause disruption. Thus, keeping only 1 feature should be tested. Examples of size-only and color-only prompts are shown here.

Size only prompt

"We have three types of colored tokens. The big one corresponds to 100, the intermediate one corresponds to 10 and the small one corresponds to 1. how to represent *number_a* with these tokens?"

Color only prompt

"We have three types of colored tokens. The green one corresponds to 100, the orange one corresponds to 10 and the black one corresponds to 1. how to represent *number_a* with these tokens?"

2.2.2 Exploring the upper bound of language models

Big Numeoristies

In the original Number Screening, most tasks contain numerosities no larger than 100. These numerosities are largely used in daily life and teaching activities. As a result, language models may be more familiar with those numerosities. To tell whether language models have mastered the laws of math or just being familiarized with those numerosities, big numerosities are introduced in the following tasks.

Parity Judgement

30 numbers consist of 10 6-digit long numbers, 10 7-digit long numbers, and 10 8-digit long numbers.

Transcoding: Arabic Numerals $\rightleftharpoons$ Words

The same 30 numbers as the Parity Judgment.

4 Basic Operation

50 pairs of numbers range from 1 to 10^5 (mostly 5-digit long numbers).

Changes of mapping: Arabic Numerals $\rightleftharpoons$ tokens

The original mapping between Arabic numerals and tokens can be found without any calculation. For instance, 145 is 1 green, 4 orange, and 5 black, so the model just has to put some digits between the words. Thus, a new mapping that requires calculation is provided.

	Basic	**Challenging**
Black-small	1	1
Green-medium	10	5
Orange-big	100	200

Table 2.1: Comparison of Two Mappings of the Transcoding Tasks

2.2.3 Order

There is no task that screens the ordering ability, so some simple tests are given to the models. Ordering may not be difficult but it is fundamental to check if the language models have a correct number line.

Ordering

30 sets of numbers, each consisting of 10 numbers that are randomly generated from -1000 to 1000, to be ordered ascendingly. 'Order these numbers: [number sets here] from small to big"

Order Judgement

30 triplets and 30 quadruples randomly generated from -100 to 100. Among them, 10 triplets/quadruples are ascending, 10 are descending, and 10 are not in order. "are these numbers:[triplets or quadruples here] in ascending or descending order, answer only yes or no"

2.3 NADL-F

The Financial Numerical Activities of Daily Living (NADL-F) assessment is a novel tool designed to evaluate financial abilities in clinical demographics(Arcara et al., 2017). Despite its relative brevity, the NADL-F effectively covers the majority of daily tasks that require financial aptitude. Psychometric analysis has confirmed its satisfactory properties, affirming its strong overall validity in the measurement of financial skills. The completed version of the NADL-F assessment is made up of seven distinct tasks. These tasks evaluate varying facets of financial capability. Each task consists of multiple items, structured in such a way as to escalate in complexity when feasible.

2.4 Counting

Counting tasks are assigned to VQA models and GPT models.

2.4.1 VQA models

Task 1: Counting the number of objects in a picture

Prompt: ”How many ’objects’ are in the picture?”

Objects will be replaced by one of the category words. There are 5 categories in total: Apples, Butterflies, Dots, Flashcards, and People. 10 numerosities ranging from 1 to 10 are tested. Each numerosity will be tested by 50 different images (stimuli); the number of objects to be counted in these images is fixed, but the position of the object itself in the picture, the object itself, and the size of the object itself will vary. Thus, 500 images are given to ViLT and BLIP separately to test their count ability.

For each category there are a few transparent PNG images; these images are resized to 200x200 if they are larger than 256x256 and then placed on a 1024x1024 white canvas randomly with the precondition that each object is at least 102 pixels away from the border and each object must not occlude each other. An example of each category with numerosity 6 is shown in Figure 2.1.

Figure 2.1: An example of stimuli for each category

Task 2: Counting the number of sides of a polygon

Prompt: ”How many sides does this polygon have?”

Polygons whose number of sides ranges from 3 to 12 are tested. Figure 2.2 shows all the polygons tested.

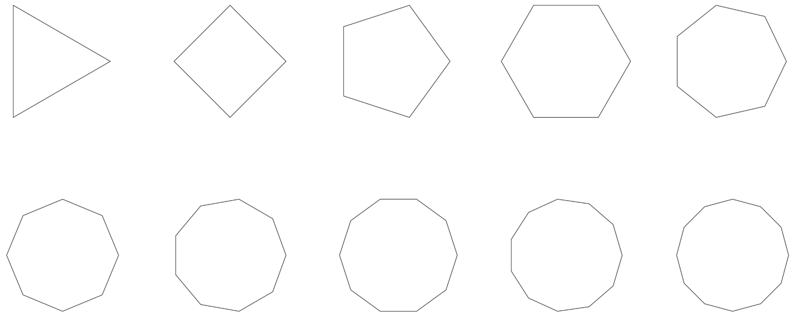

Figure 2.2: All polygons used in the test

2.4.2 GPT

GPT models are asked to count how many letters or words are in a string. Letters and words in the string can either be uniform (e.g. ’cccc’ and ’cat cat cat’) or nonuniform (e.g. ’gzxt’ and ’unicorn, helicopter, brain’).

The system input for the model is: ”Count how many letters there are. Answer with only one Arabic number.” in the letter counting task and ”Count how many words there are. Answer with only one Arabic number.” in the word counting task. The user input will later give the prompt: ”cccc”; ”gzxt gzxt gzxt”; ”hippocampus hippocampus hippocampus” or ”unicorn, helicopter, brain”.

2.5 Numerosity Generation

Numerosity Generation tasks are given to diffusion models and GPT models. Models are asked to generate a certain number of things in a text or an image.

2.5.1 Diffusion models

In this task, the prompts comprise a numerosity and a category. There are 5 categories in total: Apples, Butterflies, Dots, Flashcards, and People. Numerosity ranges from 1 to 10. Thus, one prompt could be ”7 apples”, and another could be ”1 dot”. Each numerosity is tested 50 times, thus making 2500 responses from each diffusion model.

2.5.2 GPT

GPT models are asked to generate either a string of letters or words. Letters and words in the string can either be uniform (e.g. 'cccc' and 'cat cat cat') or nonuniform (e.g. 'gzxt' and 'unicorn, helicopter, brain').

The system input for the model is: "You are an assistant". The user input will later give the prompt: "Write the word lion 5 times". in the uniform word generation task; "Write 5 different words" in the nonuniform word generation task; "Write the letter a 5 times" in the uniform string generation task; and "Write the string cbsgs 5 times" in the nonuniform string generation task.

2.6 Multimodal (visual-language) tasks

Visual-language multimodal tests consist of 2 parts. The first part tests the basic geometrical concepts of the models, and the second numerosity comparison task tests the number acuity of visual question answering models.

Geometircal Concepts Figure 2.3 shows an output of a diffusion model and Figure 2.4 shows an input for both VQA models "Draw a(n) equilateral triangle"

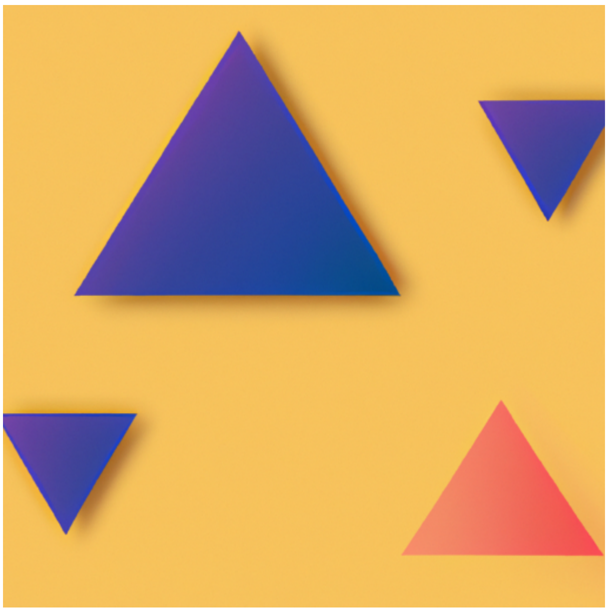

Figure 2.3: One example that DALL·E generated given the prompt

”what is this geometrical shape?”

Figure 2.4: One of the pictures used in the geometrical concept tests. A rectangle

Numerosity Comparison Vision The numerosity comparison task is a widely used tool to assess number acuity. In this task, participants are shown two numerical stimuli and asked to identify which is larger (Dehaene et al., 1990). The stimuli can be presented simultaneously or sequentially and can be represented in a variety of formats (dots in this experiment). The precision with which individuals can make these judgments is taken as a measure of their number acuity. The task is believed to be based on the innate mental number line, as it requires an estimation of the magnitude of the numbers presented and their comparison (Halberda et al., 2008).

The numerosity comparison task consists of 210 trials. Each trail contains one image. Each image has two black square frames, one on the left and the other on the right. One of them is the target, the target ranges from 1 to 12 and the other is the reference, which ranges from *target-5* to *target+5* by one. The lower limit is set to be a positive number. Half of the trails have the target on the left and the other on the right. Fiure 2.5 and Figure 2.6 show an example of the visual inputs when the size and numerosity are congruent and when the condition is incongruent respectively.

Congruent where the side has greater magnitudes also has larger dot size

Incongruent where the side has smaller magnitudes has larger dot size.

”which side has more dots?”

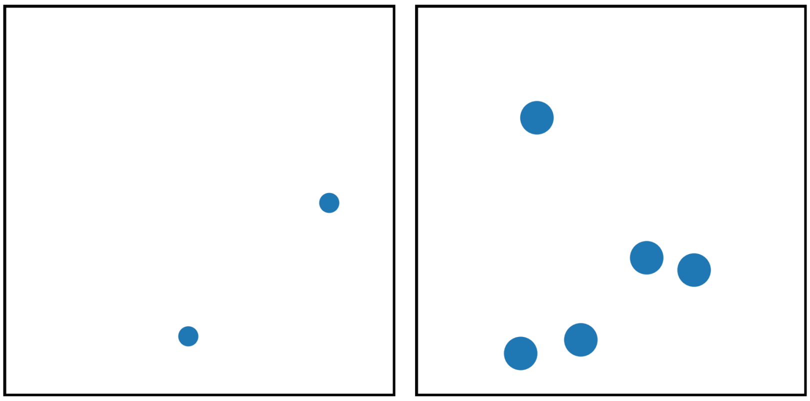

Figure 2.5: Example trail of numerosity comparison with dots. Condition: size and numerosity are congruent

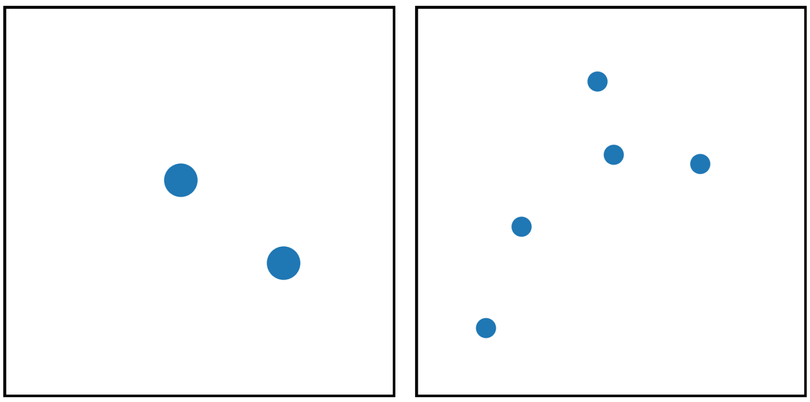

Figure 2.6: Example trail of numerosity comparison with dots. Condition: Size and numerosity are incongruent

2.7 Models

2.7.1 GPT-3.5

GPT-3.5, an advancement of the Generative Pretrained Transformer (GPT) series developed by OpenAI (OpenAI, 2022a), has demonstrated impressive capabilities in a wide range of language tasks(Koubaa, 2023). It has been an object of extensive research due to its exceptional performance in natural language processing (NLP) and machine learning tasks.

At the structural level, GPT-3.5 adopts the transformer-based architecture as its predecessors, utilizing self-attention mechanisms to capture complex patterns in the data. This structure allows the model to effectively handle a sequence of data input and output, facilitating context understanding and prediction generation.

GPT-3.5 training is a two-step process: pre-training and fine-tuning. In the pre-training phase, GPT-3.5 learns to predict the next word in a sentence by training on a diverse range of internet text. However, the model has no access to specific documents or sources. Once the base model is pre-trained, it is fine-tuned on a narrower data set, generated with the help of human reviewers following specific guidelines provided by OpenAI.

GPT-3.5 has several key features that distinguish it from previous models. First, it has an impressive language generation capability, where the model generates human-like text that is contextually relevant and syntactically correct. Second, GPT-3.5 exhibits a better understanding of the context, thus improving the relevance and appropriateness of its responses. Third, it has a high-level capability to adapt to new prompts and generate accurate responses.

In conclusion, GPT-3.5 is a significant contribution to the field of NLP and AI, pushing the boundaries of what is possible in the realm of machine intelligence and opening up new opportunities for research and applications.

2.7.2 GPT-4

GPT-4, the latest incarnation in the Generative Pretrained Transformer series by OpenAI, represents a remarkable leap in the field of natural language processing (NLP) and artificial intelligence (AI). It has continued to draw significant attention in the academic community due to its robust capabilities in language understanding, generation, and broader machine learning tasks(OpenAI, 2023).

In terms of structure, GPT-4 maintains the transformer-based model architecture, analogous to its predecessors. This architecture uses self-attention mechanisms, which allow the model to handle sequential data inputs effectively, better understand the context, and generate accurate predictions. The capacity of GPT-4 has been considerably expanded, leading to improved performance in various language tasks.

GPT-4 follows a similar training regimen to GPT-3.5, comprised of pre-training and fine-tuning phases. During pre-training, the model is trained to predict the next word in a sentence using a vast and diverse corpus of internet text. The exact documents or sources used in this phase remain anonymous to the model. The subsequent fine-tuning phase hones the model's abilities on a narrower data set generated with human reviewers' help, following OpenAI's specific guidelines.

Several key features distinguish GPT-4. Its language generation abilities have been significantly improved, with the model generating text that closely emulates human language in terms of context, syntax, and relevance. Furthermore, GPT-4 showcases improved understanding and contextual relevance, enhancing the precision and appropriateness of the responses it generates. The model also demonstrates high adaptability to new prompts, generating more accurate responses.

Despite these advances, GPT-4, like all AI models, has its limitations, including potential biases derived from training data and occasional generation of non-sensical or incorrect output. Ongoing research aims to alleviate these limitations, further enhancing the utility and efficiency of the model.

In conclusion, GPT-4 presents a significant breakthrough in the NLP and AI domains, pushing the boundaries of machine intelligence capabilities. It offers exciting new avenues for research and real-world applications that promise to revolutionize our interaction with technology.

2.7.3 ViLT

The Vision-and-Language Transformer (ViLT) is a novel model that aims to simplify and improve the efficiency of Vision-and-Language Pre-training (VLP)(Kim et al., 2021). The main innovation of ViLT lies in its unified approach to processing both visual and textual input.

Unlike traditional models in this field, which often rely heavily on convolutions and object detection for feature extraction from images, ViLT treats visual inputs in the same convolution-free manner as textual inputs. This drastically simplifies the pre-processing steps and leads to a significant boost in efficiency and speed. It has been shown to be up to several tens of times faster than previous VLP models.

The architecture of ViLT is based on the Transformer model, which has been highly successful in Natural Language Processing (NLP) and is known for its self-attention mechanism. By leveraging the Transformer architecture, ViLT can process visual and textual input concurrently in a unified framework, allowing for richer interactions and fusion of information between the two modalities.

Furthermore, ViLT also addresses the limitation of expressive power in previous VLP models, which is often capped by the expressive power of the visual embedder and its predefined visual vocabulary. By processing visual input in a similar manner to textual input, ViLT can potentially capture a wider range of visual features and concepts, thus expanding its expressive power. Despite its simplicity and efficiency, ViLT achieves competitive or even superior performance in various downstream tasks, demonstrating its effectiveness and potential as a new approach to VLP.

2.7.4 BLIP

Advancements in Vision-Language Pre-training have significantly improved the outcomes for various vision-language tasks. Despite this, the majority of pre-existing models have excelled either in tasks requiring understanding or those involving generation. Moreover, most of the performance enhancement has been primarily obtained by enlarging the data set with image-text pairs sourced from the web, which often contain noise and are less than ideal for supervision.

In response to these limitations,Li et al., 2022 introduced BLIP, a novel VLP framework designed to flexibly transition to tasks that involve the understanding and generation of vision-language. BLIP optimally harnesses the noisy data sourced from the Web through a process that we term ”bootstrapping the captions.” This process involves the generation of synthetic captions by a captioner, followed by the removal of noisy ones by a filter.

The implementation of BLIP has resulted in state-of-the-art outcomes across a broad array of vision-language tasks. These include image text retrieval, where we have observed an average recall increase of 2.7%, image captioning, with a CIDEr increase of 2. 8%, and visual question answering (VQA), where we observed a VQA score increase of 1.6%. Moreover, BLIP has demonstrated robust generalization capabilities, especially when directly transferred to video-language tasks in a zero-shot context.

2.7.5 DALL·E

DALL·E, another innovative creation of OpenAI, is a remarkable model that expands the application of artificial intelligence beyond the realm of natural language processing and into the world of image synthesis(OpenAI, 2022b).

Built on the foundations of GPT-3, DALL·E incorporates the powerful transformer-based architecture that is a staple in OpenAI’s generative models. However, DALL·E extends this by not just processing text, but also producing images from textual descriptions, illustrating a significant step in multi-modal AI models. It incorporates an additional component called VQ-VAE-2, a form of autoencoder that is used to generate high-quality images.

The training process for DALL·E closely follows the two-step approach similar to GPT-3.5 and GPT-4: pretraining and fine-tuning. During pretraining, DALL·E is trained on a large data set that contains pairs of textual descriptions and the corresponding images. This data set does not contain specific information about the documents or sources, preserving the model’s ability to generalize over various inputs. In the fine-tuning phase, the model is trained on a narrower data set, helping it to create highly specific and detailed images from textual prompts.

DALL·E’s ability to generate detailed and contextually relevant images from textual descriptions is its key feature. The model showcases a deep understanding of language and the associated visual concepts, which makes it capable of generating creative renditions of described scenarios. Its applications range from generating unique artwork to aiding in design processes across various industries.

2.7.6 Stable Diffusion

Stable Diffusion (SD) is a latent text-to-image diffusion model trained on 512x512 images from a subset of the LAION-5B database. Following a similar approach as Google's Imagen, this model applies a text encoder from CLIP ViT-L/14, which is kept in a 'frozen' state, to shape the model based on text prompts. The model is notably efficient, with a UNet of 860M and a 123M text encoder, making it relatively compact (Rombach et al., 2022).

Latent diffusion models (LDMs) excel in several areas such as unconditional image generation, inpainting, and super-resolution, displaying commendable performance. They offer the added advantage of substantially lower computational demands compared to pixel-based diffusion models.

Versions of Stable Diffusion

Stable Diffusion v1 series denotes a unique setup of the model structure that utilizes an autoencoder with a downsampling-factor of 8, alongside an 860M UNet and a CLIP ViT-L/14 text encoder for the diffusion process. Initially, the model is preprocessed using 256x256 resolution images, which is subsequently fine-tuned using higher resolution 512x512 images.

Stable Diffusion v2 series denotes a particular arrangement of the model structure, incorporating an autoencoder with a downsampling-factor of 8, an 865M UNet, and an OpenCLIP ViT-H/14 text encoder for the diffusion model. The SD 2-v model variant is capable of generating outputs with a resolution of 768x768 pixels.

2.7.7 Specs and Tasks of the Models

The following table shows the versions of the models (if more than one are provided), the specs of the models, and the tests applied. Only specs that are not in default are mentioned here.

Neural Network	Specs	Tests
GPT-3.5[Version Jan30] & GPT-4	Max_token=[20,200], temperature=0.7	Number Screening & Pure linguistic tasks of NADL-F
ViLT	All Default	Visual Question Answering
BLIP	Reasoning at single precesion	Visual Question Answering
DALL·E[Version 2022Jan]	size='512x512', '1024x1024'	Generating geometrical shapes and numbers of objects
Stable Diffusion v1-5, Stable Diffusion v2-1, Stable Diffusion v2-2_XL	size='512x512', inference steps= 50, guidance scale=8.5	Generating geometrical shapes and numbers of objects

Table 2.2: Specs of Models and Where the Models Are Tested

For commercial models like DALL·E, GPT-3.5, and GPT-4, OpenAI's and StabilityAI's API services were used. Open-source models like Stable Diffusion v1-5, v2-1, and ViLT, models were run locally with precision at fp16. The detailed Python Jupyter Notebook can be accessed from the Appendix. An example is also listed in the Appendix.

Results

3.1 Number Screening

GPT-4 shows almost perfect performance in all basic number screening tasks, while GPT-3.5 failed dramatically in some tricky tasks, although for humans the answers to those tasks are trivial. Accuracies of both models with respect to different tasks in Number Screening are shown in Figure 3.1.

3.2 Customized data set

In general, GPT-4 can solve most challenging questions except for the multiplication of 2 big numerosities. On the other hand, GPT-3.5 only answers the parity judgment of big numerosities correctly. In the end, we can conclude that GPT-3.5 is bad at the manipulation of big numerosities. Accuracies of both models concerning different tasks are shown in Figure 3.2. For deconstructed tasks, GPT-3.5 performs better in the transcoding token to Arabic numerals but worse in the transcoding Arabic numerals to token tasks. It seems that transcoding Arabic numerals to tokens, for GPT-3.5, at least 2 features on the target toke to help GPT-3.5 calculate.

3.2.1 Disclosure of complex tasks

Deconstructed tasks On the other hand, reducing the reasoning steps or the degree of complexity of the task does not help GPT-3.5 to perform better or to do so, the precision is identical to each other. Combining the response example shown above, it is reasonable to presume that both GPT-3.5 and GPT-4 grasp the key information and the method to solve this task correctly;

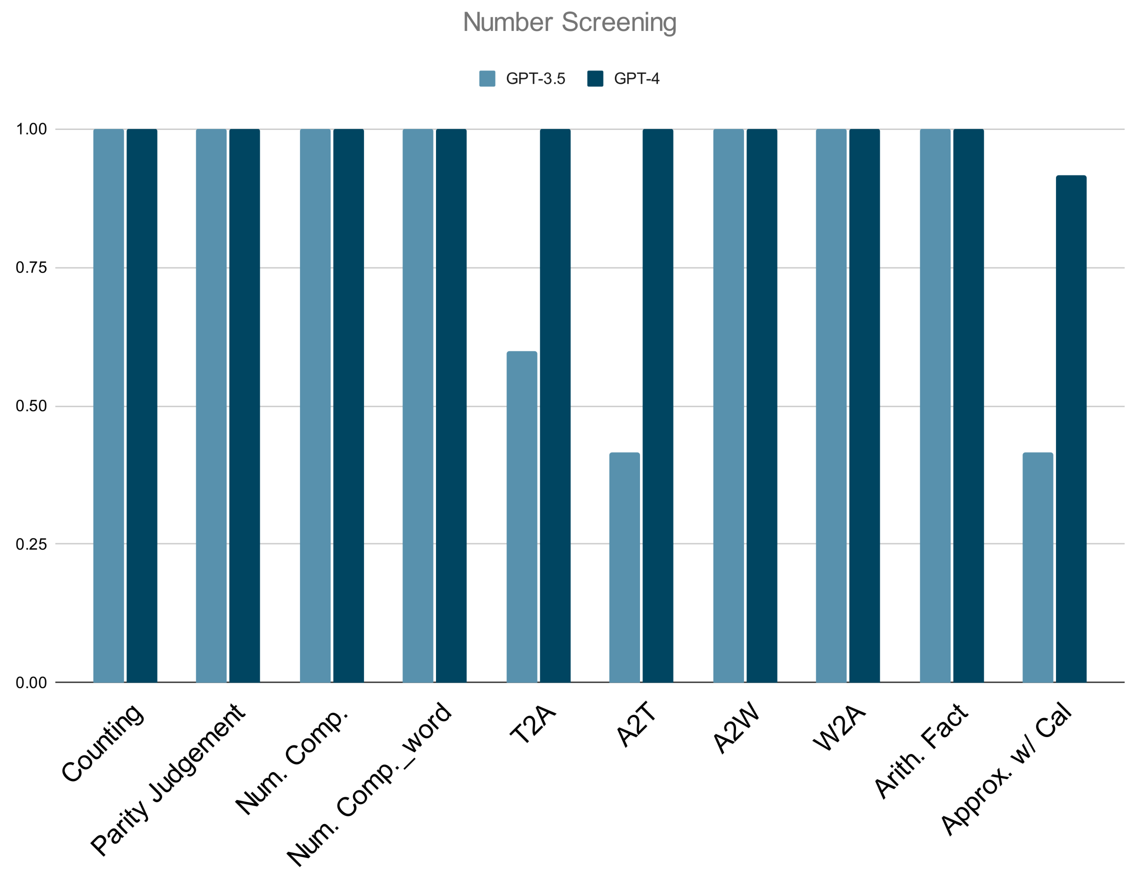

Figure 3.1: Results of Number Screening standard tests by GPT models
T2A: tokens to arabic numerals; A2T: arabic numerals to tokens; Num. Comp.: Number Comparison; W2A: word to arabic numerals; A2W: arabic numerals to word; Approx. w/o Cal: Approximation without calculation of an operation; Approx. w/ Cal: Approximation with the calculation of an operation

however, the accuracy of the purely mathematical calculation ability is not the same across the models.

Example in the Prompt The prompts for the poorly performed tasks of GPT-3.5 are redesigned by adding an example to the original prompt. These tasks are transcoding between Arabic numerals and tokens; and approximation with calculation. After adding a simple example, the performance of GPT-3.5 improved greatly and can even match the performance of GPT-4 to some extent (see Figure 3.3.

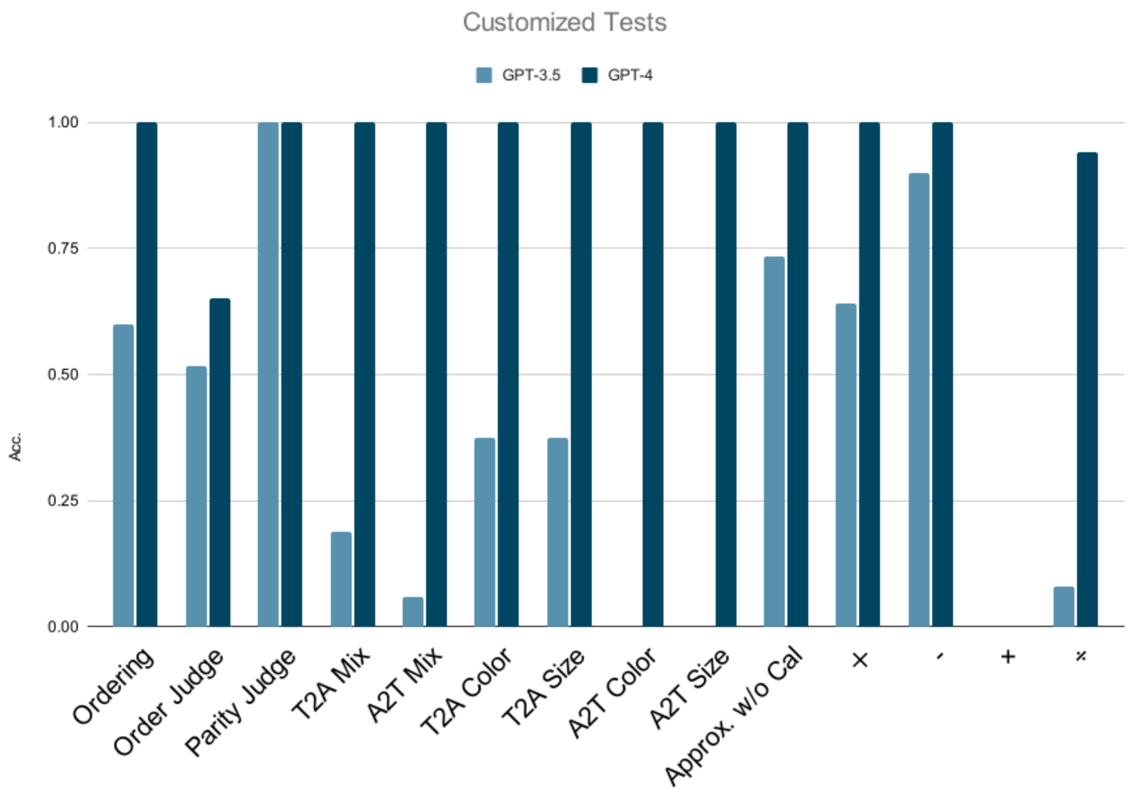

Figure 3.2: Results of customized Number Screening tests by GPT models
T2A: tokens to arabic numerals; A2T: arabic numerals to tokens; Approx. w/o Cal: Approximation without calculation of an operation; +: addition; -:subtraction; ×:multiplication; ÷: division

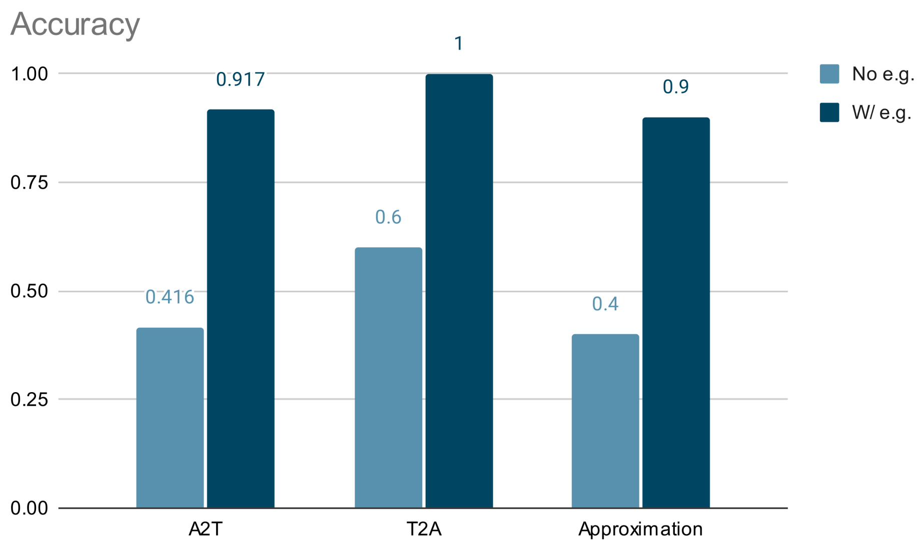

Figure 3.3: Results comparing the 2 different prompts

T2A: tokens to Arabic numbers; A2T: Arabic numbers to tokens

No e.g.: no example of is given in the prompt; W/ e.g.: 1 example of solving this problem is given in the prompt together with the question

3.2.2 Exploring the upper bound of language models

Big Numerosites More challenging tasks do not affect the final performance of GPT-4. As for GPT-3.5, except for the Parity Judgement task, the models are affected by the increased length of sequences (i.e., big numerosities). In the end, we can conclude that GPT-3.5 is bad at the manipulation of big numerosities but GPT-4 can handle big numerosities that are about 10-digit long.

Changes of Mapping In this task, after changing the mapping relation, GPT-4 still managed to perform perfectly, while GPT-3.5 failed to deliver good performance. An example trial is shown here for comparison.

Prompt: "We have three types of colored tokens. The big green one corresponds to 200, the intermediate orange corresponds to 5, and the small black one corresponds to 1. how to represent 88988 with these token(s)? try to use as less token as possible"

An example of responses after changing the mapping
GPT-4 Response: "To represent the number 88988 using the colored tokens (green, orange, and black) with the least number of tokens possible, you can follow these steps: 1. Determine the maximum number of green tokens (each worth 200) that can be used without exceeding 88988. 2. Calculate the remaining value after using the green tokens. 3. Determine the maximum number of orange tokens (each worth 5) that can be used without exceeding the remaining value. 4. Calculate the remaining value after using the orange tokens. 5. Use black tokens (each worth 1) to represent the remaining value. Let's go through the calculations: 1. Green tokens: 88988 ÷ 200 = 444.94 → Use 444 green tokens (since you can't use a fraction of a token). 2. Remaining value after green tokens: 88988 - (444 × 200) = 88988 - 88800 = 188 3. Orange tokens: 188 ÷ 5 = 37.6 → Use 37 orange tokens (since you can't use a fraction of a token). 4. Remaining value after orange tokens: 188 - (37 × 5) = 188 - 185 = 3 5. Black tokens: Use 3 black tokens to represent the remaining value of 3. So, to represent the number 88988 using the least number of tokens, you'll need 444 green tokens, 37 orange tokens, and 3 black tokens." **GPT-3.5 Response:** "Green token: 444 Orange token: 1777 Black token: 88"

Table 3.1: Comparison between GPT-3.5 and GPT-4 after changing the mapping

3.2.3 Order

In the ordering task, GPT-4 maintains its perfect performance. GPT-3.5 orders about 60% of trials correctly. Both models cannot tell if a few numbers are in order. GPT-3.5 performs at the chance level and GPT-4 performs slightly better (see Figure 3.2).

3.3 NADL-F

Similar to the Number Screening tasks, GPT-4 performed perfectly for the linguistic part of the NADL-F while GPT-3.5 failed in those tasks that require reasoning and calculation (see Figure 3.4.

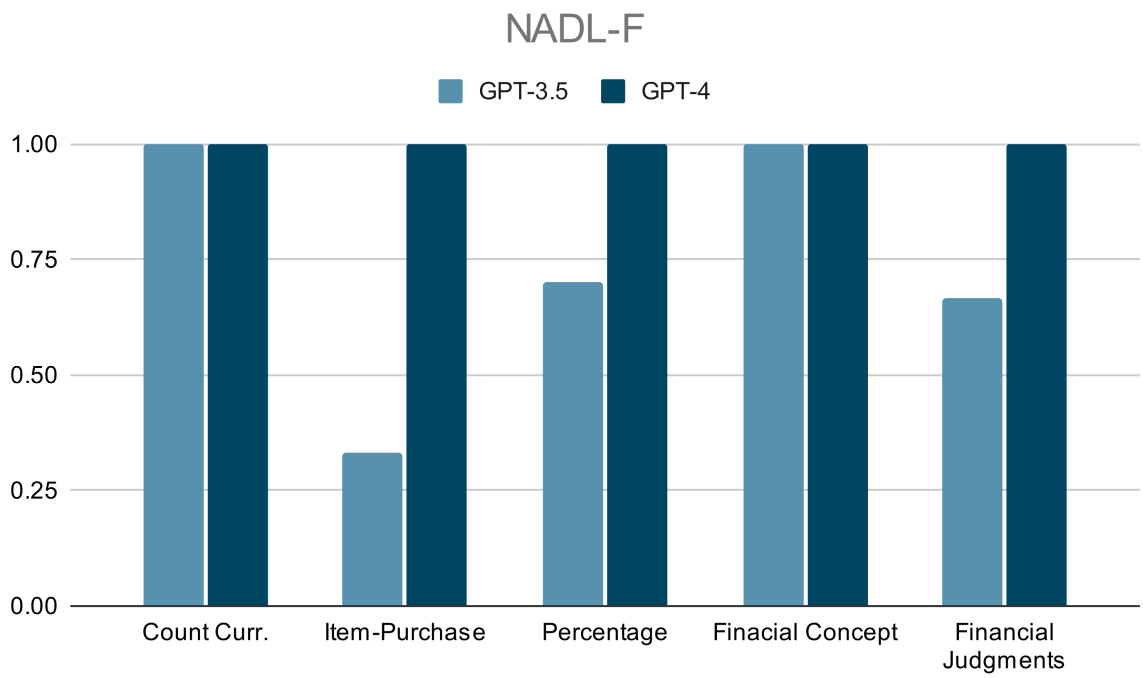

Figure 3.4: Results of NADL-F by GPT models, the vertical axis represents the accuracy Count CUrr.: Count Currencies.

3.4 Counting

Data Processing GPT models and VQA models give textual responses, so models may answer with words like 'one', 'two', 'three' or even a sentence instead of Arabic numerals like '1', '2', '3'. In this case, the response will be manually transcoded to Arabic numerals to facilitate subsequent data processing.

Confusion matrices of each generated numerosity against the target numerosity are created. Within these 20 by 10 matrices, each cell will display the number of generated numerosity, and shades of color are used to indicate relative frequency. The deeper the blue, the higher the relative frequency.

Responses where the numerosity is larger than 20 will be excluded for the generation of the confusion matrices and relative frequencies. However, when calculating accuracy, those responses are regarded as false responses rather than being excluded.

3.4.1 VQA models

ViLT and BLIP are tested. In general, BLIP has better count ability across all 5 categories.

Apples

Accuracy The overall accuracy for ViLT and BLIP is 0.158 and 0.408, respectively. Figure 3.5 shows the accuracies for each numerosity of both models.

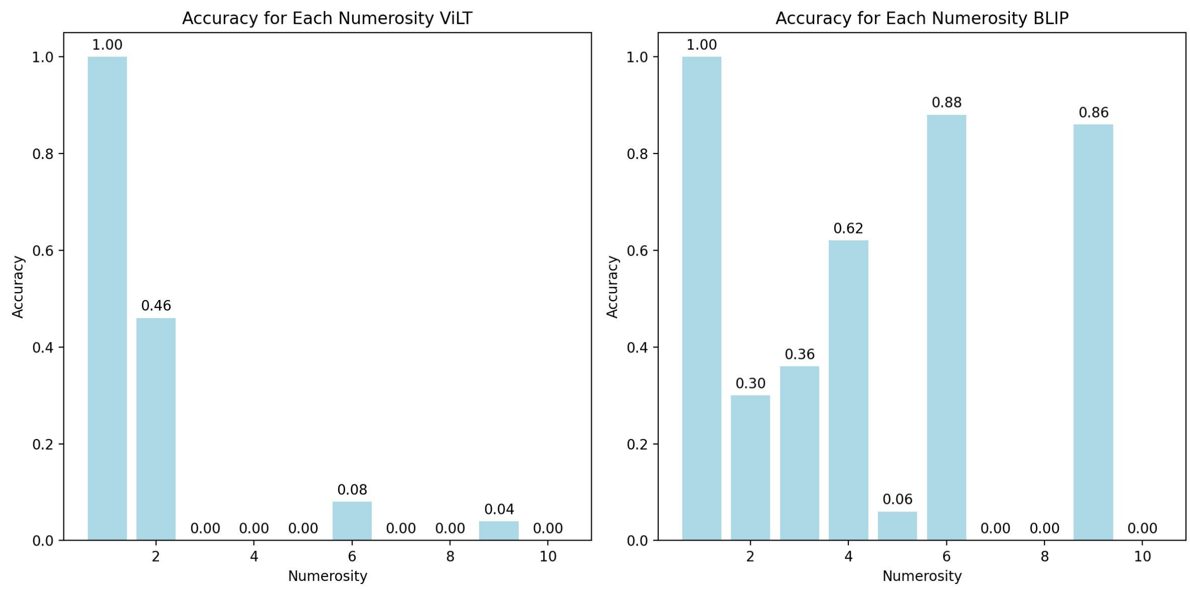

Figure 3.5: comparison between ViLT (left) and BLIP (right) in terms of accuracy where the category is Apple. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix As shown in Figure 3.6 both models can count numerosity 1 perfectly. As the numerosity goes larger and larger, the number of apples counted by ViLT is lower than the actual number of objects. For BLIP, it is more accurate and the counted number lies on both sides of the diagonal line of the confusion matrix. Checking the relative frequency of generated numerosity we can conclude that ViLT responds with a numerosity that is smaller than 7 while BLIP responds with numerosities from 1 to 10 more evenly.

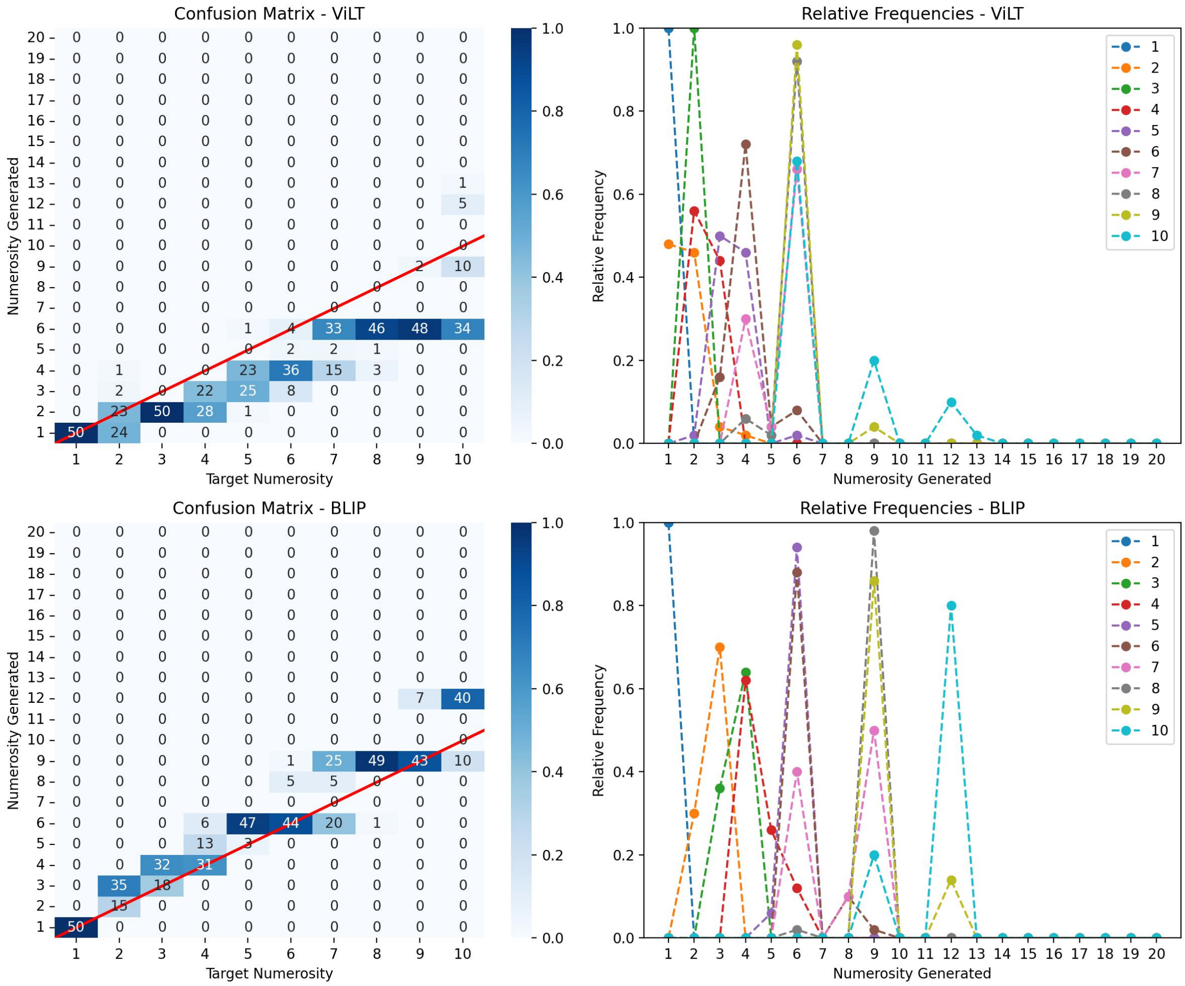

Figure 3.6: Comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the category is Apple. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Butterflies

Accuracy The overall accuracy for ViLT and BLIP is 0.436 and 0.472, respectively. Figure 3.7 shows the accuracies for each numerosity of both models.

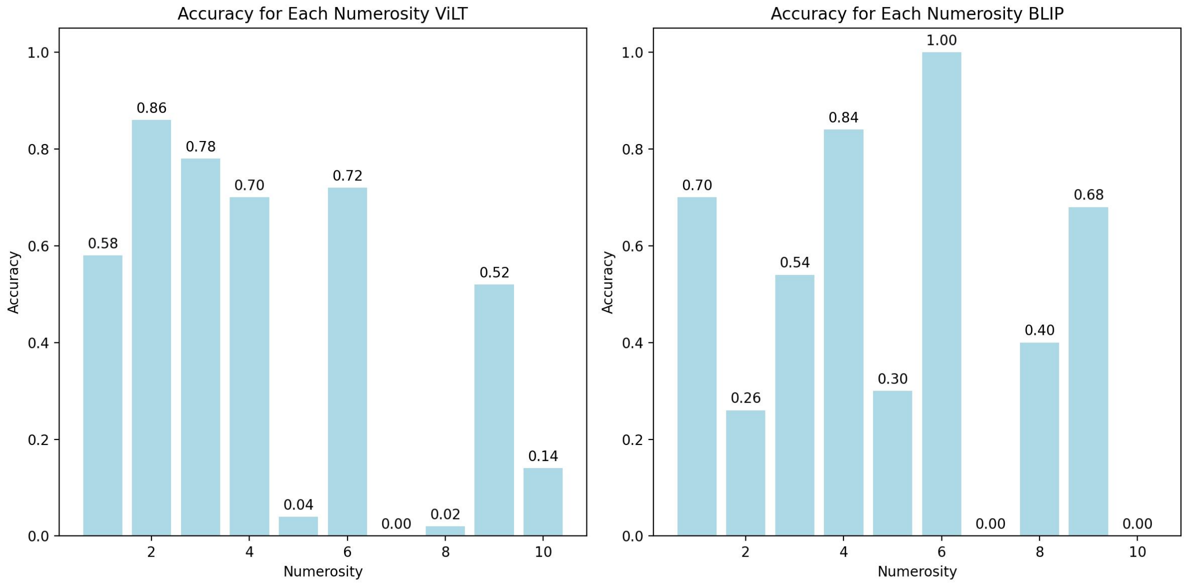

Figure 3.7: comparison between ViLT (left) and BLIP (right) in terms of accuracy where the category is Butterfly. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix ViLT still possesses the tendency to generate numerosity that is smaller than 7. BLIP, when tested by butterflies, is obsessed with numerosity 6. When prompted with an image whose number of butterflies ranges from 5 to 8, BLIP tends to respond with 6, most of the time (see Figure 3.8).

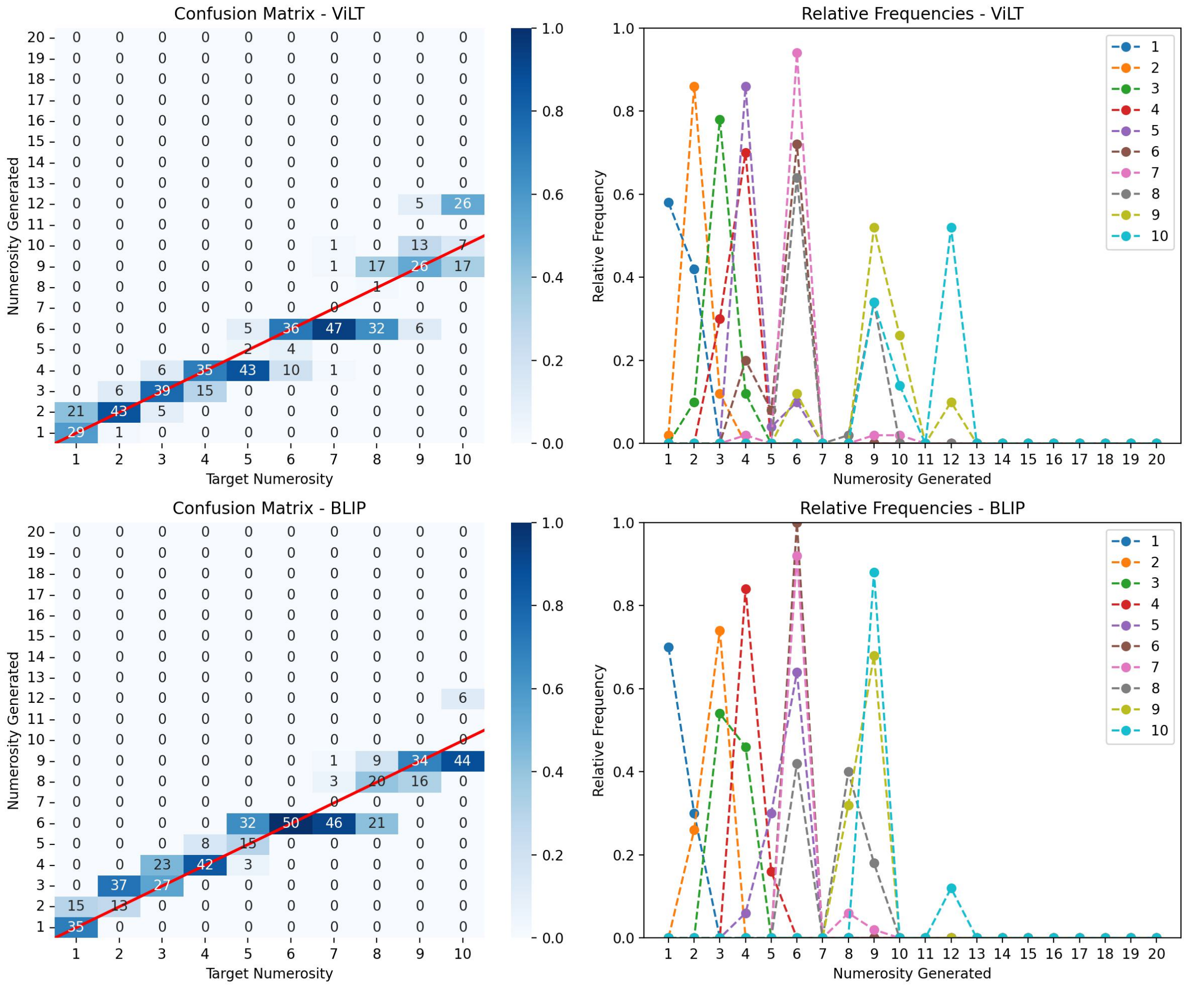

Figure 3.8: comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the category is Butterfly. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Dots

Accuracy The overall accuracy for ViLT and BLIP is 0.050 and 0.098, respectively. Figure 3.9 shows the accuracies for each numerosity of both models.

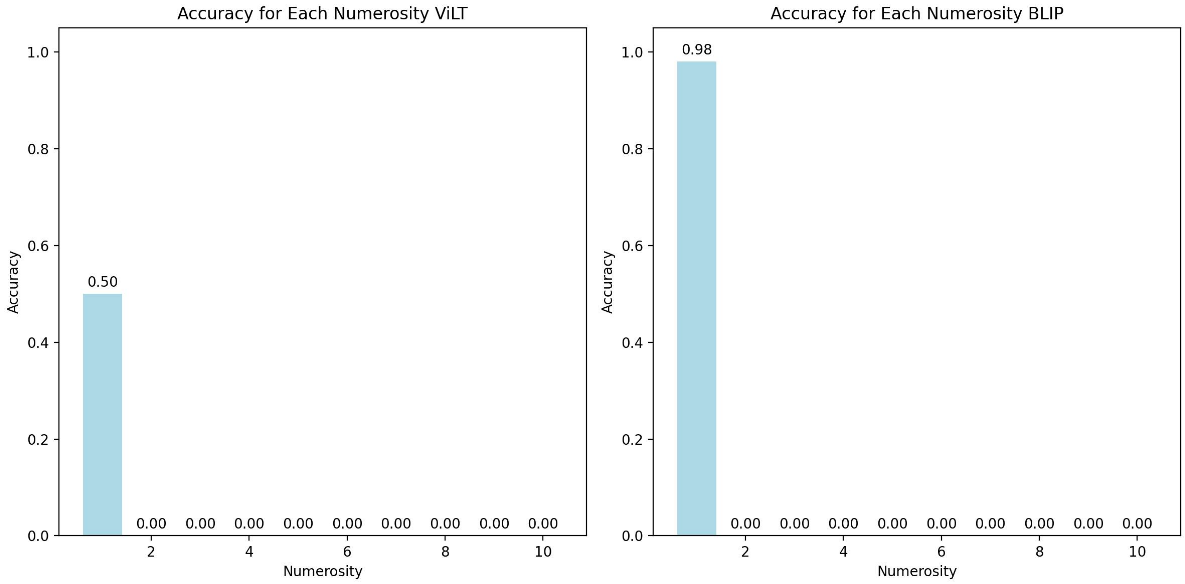

Figure 3.9: comparison between ViLT (left) and BLIP (right) in terms of accuracy where the category is Dot. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix As we can see in Figure 3.10, both models show great room for improvement when counting dots. BLIP responds to 1 dot stimuli almost perfectly but once numerosity goes beyond 1 it starts and cannot make a single correct response. Similar situation for ViLT, where ViLT did worse even when the numerosity is 1. In terms of distance to target numerosity, ViLT's responses distribute further from the diagonal redline compared with BLIP.

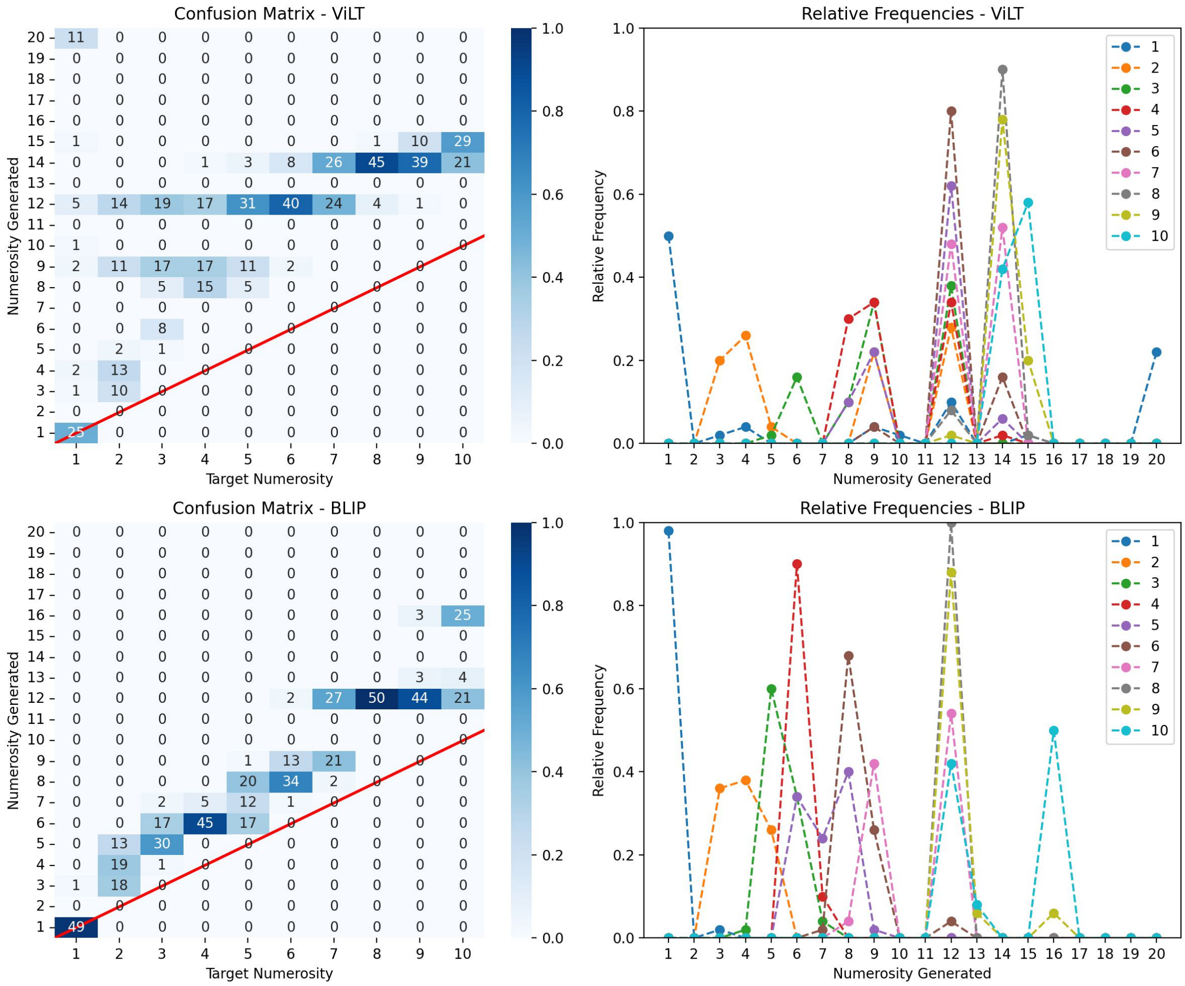

Figure 3.10: Comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the category is Dot. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Flashcards

Accuracy The overall accuracy for ViLT and BLIP is 0.092 and 0.522, respectively. Figure 3.11 shows the accuracies for each numerosity of both models.

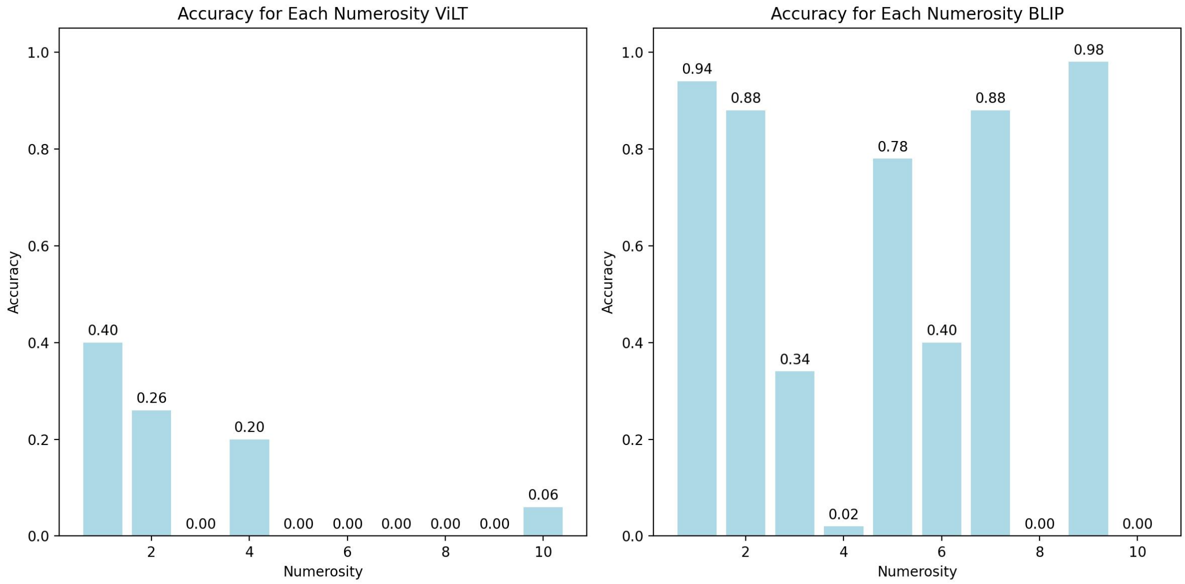

Figure 3.11: comparison between ViLT (left) and BLIP (right) in terms of accuracy where the category is Flashcard. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix ViLT barely responds with numerosity larger than 4 (see Figure 3.12). BLIP delivered a fair enough performance in general. However, if we check each numerosity's responses, BLIP can definitely perform better on numerosity 4, 8, and 10.

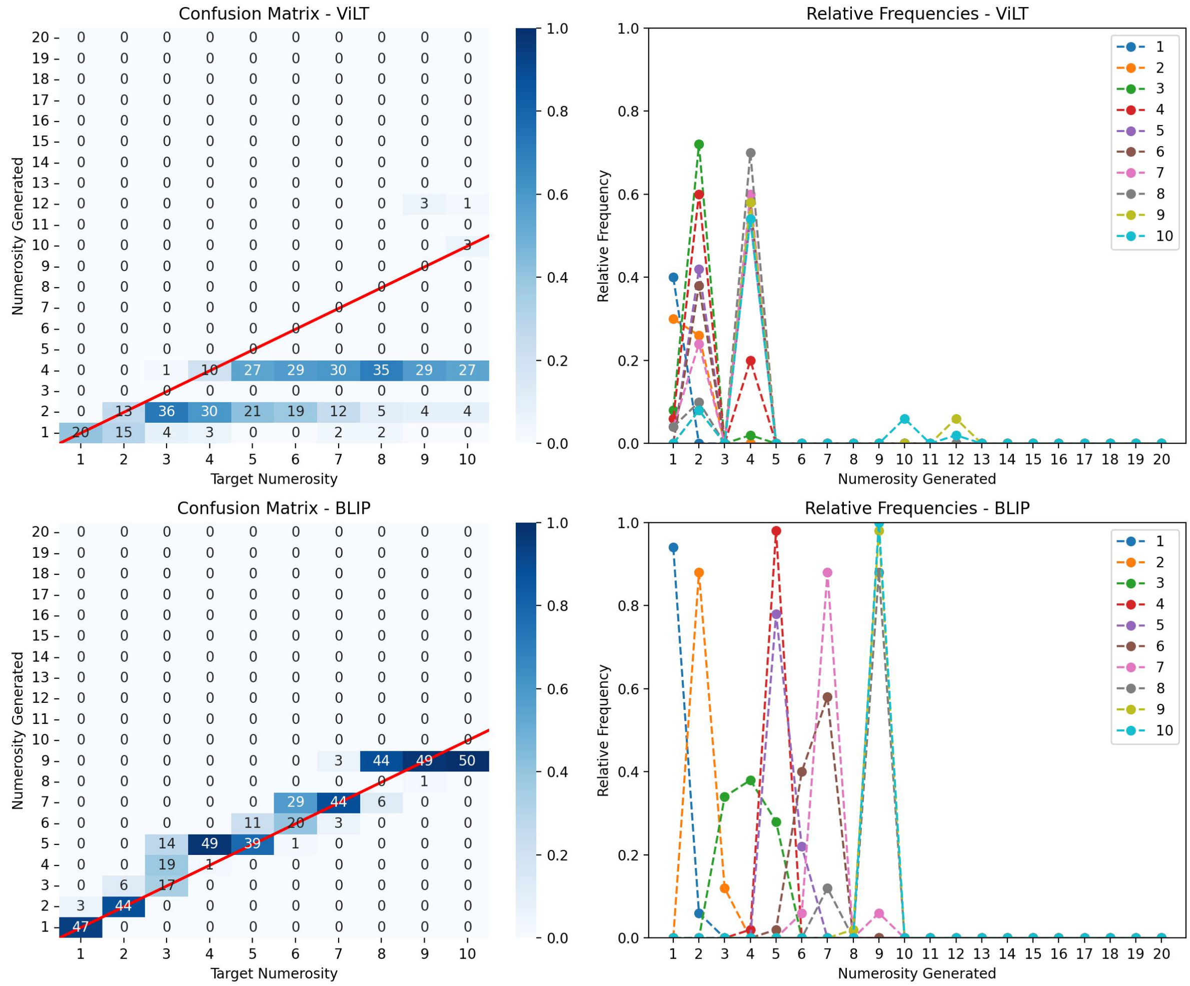

Figure 3.12: comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the category is Flashcard. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

People

Accuracy The overall accuracy for ViLT and BLIP is 0.270 and 0.504, respectively. Figure 3.13 shows the accuracies for each numerosity of both models.

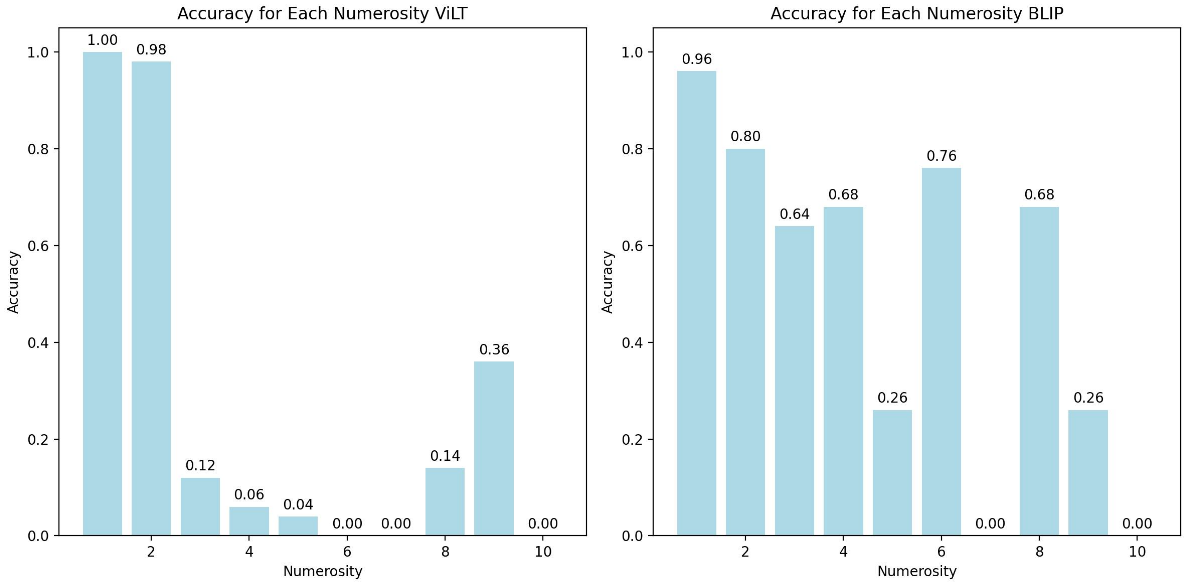

Figure 3.13: comparison between ViLT (left) and BLIP (right) in terms of accuracy where the category is People. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix Both models count the first 2 numerosity well enough. As the numerosity gets larger, ViLT drops its performance greater than BLIP. Similar trends can be found with the Apple category where ViLT tends to count less and BLIP has a more evenly distributed response in general (see Figure 3.14).

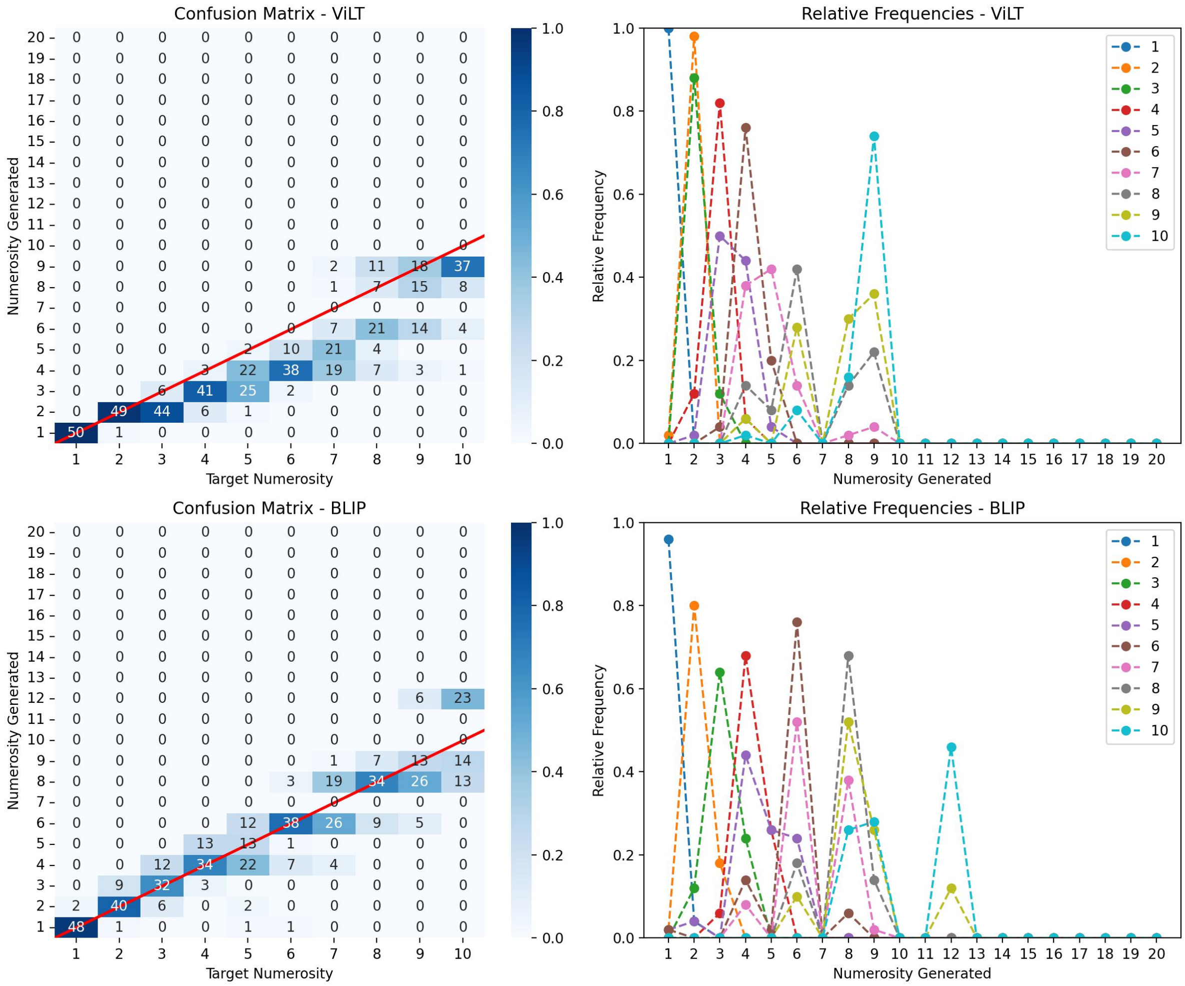

Figure 3.14: comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the category is People. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

3.4.2 GPT Models

GPT-3.5 and GPT-4 are tested. GPT-4 shows an almost perfect count ability. GPT-3.5 also gives well enough responses.

Letter non-uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.798 and 1.0, respectively. Figure 3.15 shows the accuracies for each numerosity of both models.

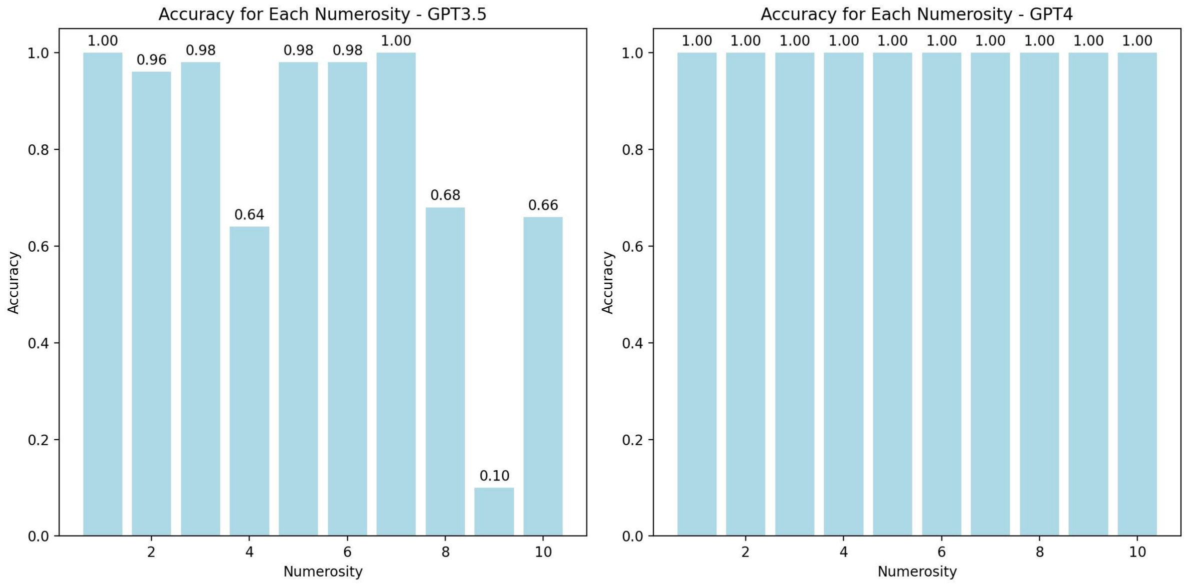

Figure 3.15: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is letter non-uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix GPT-4 counts the non-uniform strings without any mistakes. GPT-3.5 makes mistakes on numerosity 4; 8; 9; 10.

Taking a closer look at the trials that GPT-3.5 fails, we cannot say that GPT-3.5 does it wrong 100 percent. For numerosity 4, where the string to be counted is ”ccpn”, as shown in Figure 3.16, GPT-3.5 regards this as a 3-letter string. It makes sense since the prompt given is

”Count how many letters there are. Answer with only one Arabic number”

rather than

”Count the number of letters in this string. Answer with only one Arabic number”

However, this is not always the case. As for numerosity 8, where the string is ”subcyrry”, GPT-3.5 counts it as a 7-letter string rather than a 6-letter string among those false responses. It could also be the reason that when 2 same letters consequently occur in a non-uniform string, GPT-3.5 may count these 2 letters as 1 letter.

It is worth noting that GPT-4 has no such tendency; when the string to be counted is ”rss”, GPT-4 still thinks that it is a 3-letter string.

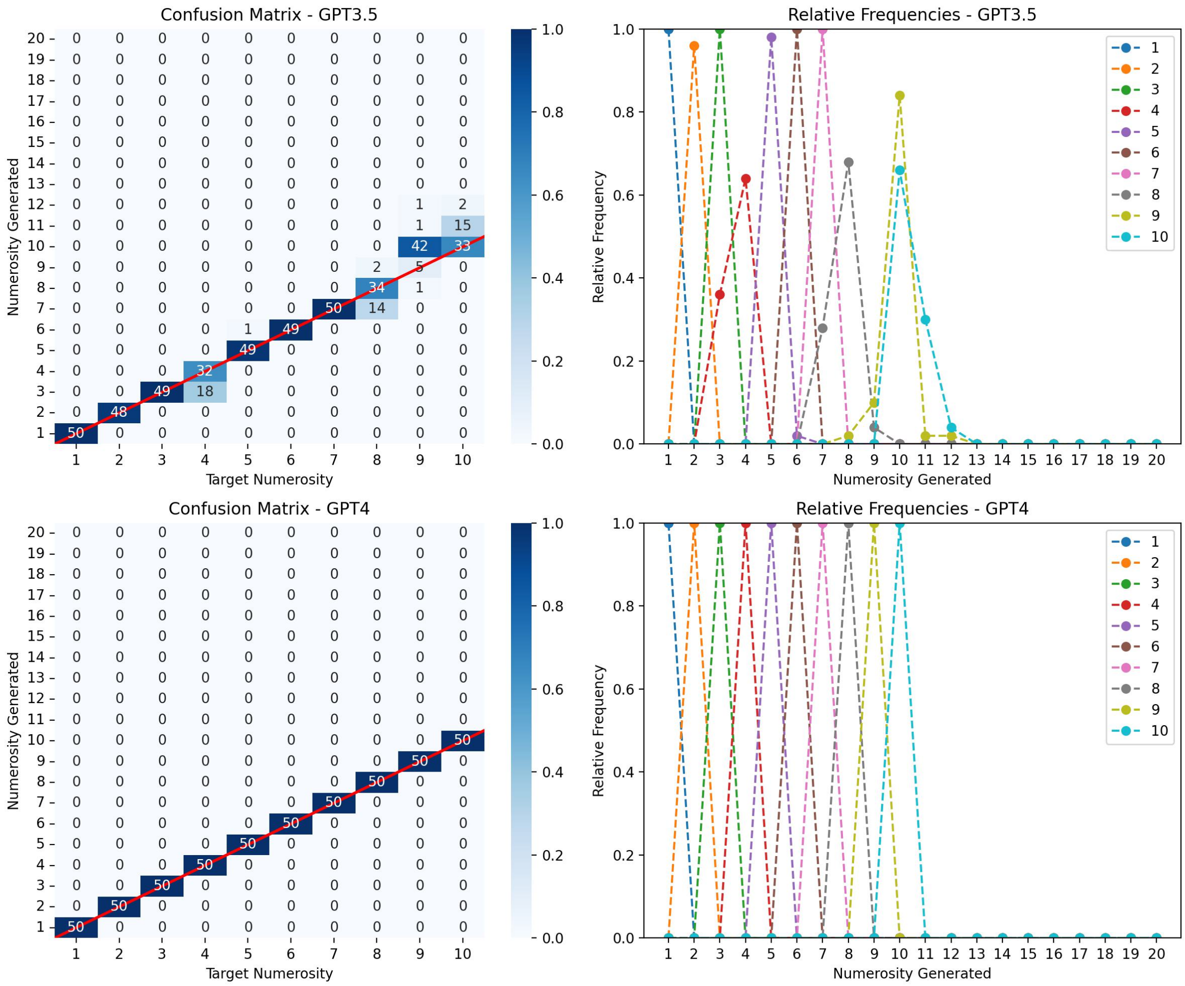

Figure 3.16: comparison of the confusion matrix and relative frequency between GPT-3.5 (top) and GPT-4 (bottom) where the condition is letter non-uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Letter uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.884 and 0.998, respectively. Figure 3.17 shows the accuracies for each numerosity of both models.

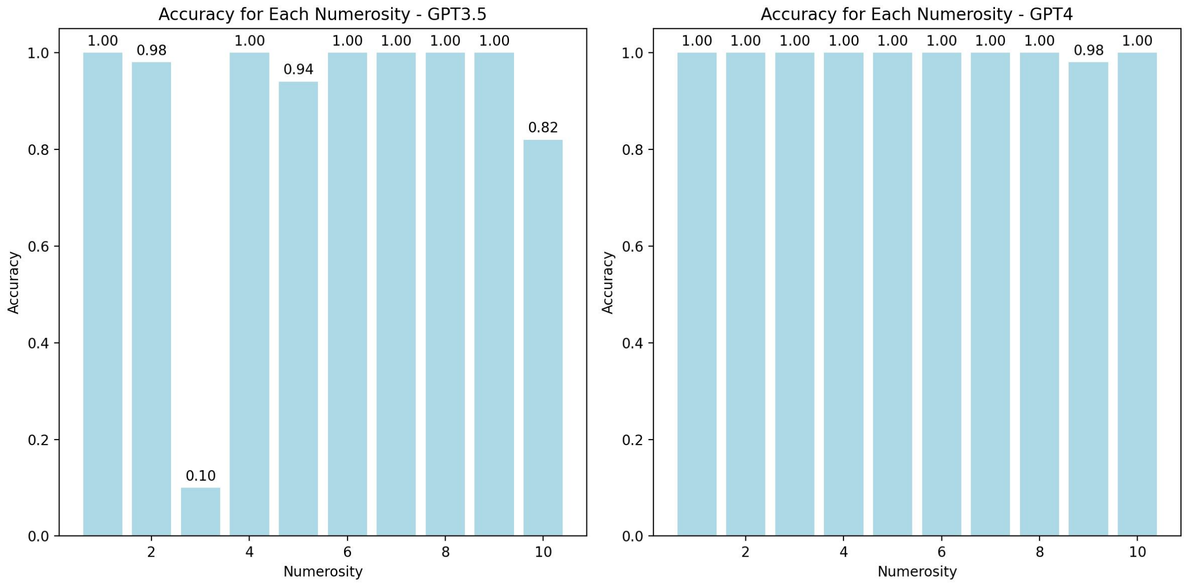

Figure 3.17: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is letter uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix GPT-4 makes only 1 mistake among 500 trails, where it counts a 9-letter string as an 8-letter string. GPT-3.5, on the other hand, only makes minor mistakes among most numerosities but can barely count a 3-letter uniform string, which is very bizarre (see Figure 3.18. The uniform string is ”www”, it could be the reason that ”www” has been tokenized as one integrity with GPT-3.5 so deeply that when it comes to ”www”, GPT-3.5 thinks the start of a domain name (e.g. www.unipd.it).

As a result, the other four 3-letter uniform strings are tested for 50 trials each; they are ”aaa”; ”ccc”; ”zzz”; ”ggg”. The accuracy for ”aaa”; ”ccc”; ”ggg” is 1.0, but for ”zzz”, there are still 28 out of 50 times that GPT-3.5 counts it as a 0-letter string. This suggests that GPT-3.5 does have the ability to count uniform strings that are 3 letters long. However, for some certain letters, it cannot count correctly, and the reason behind this is beyond the scope of this thesis.

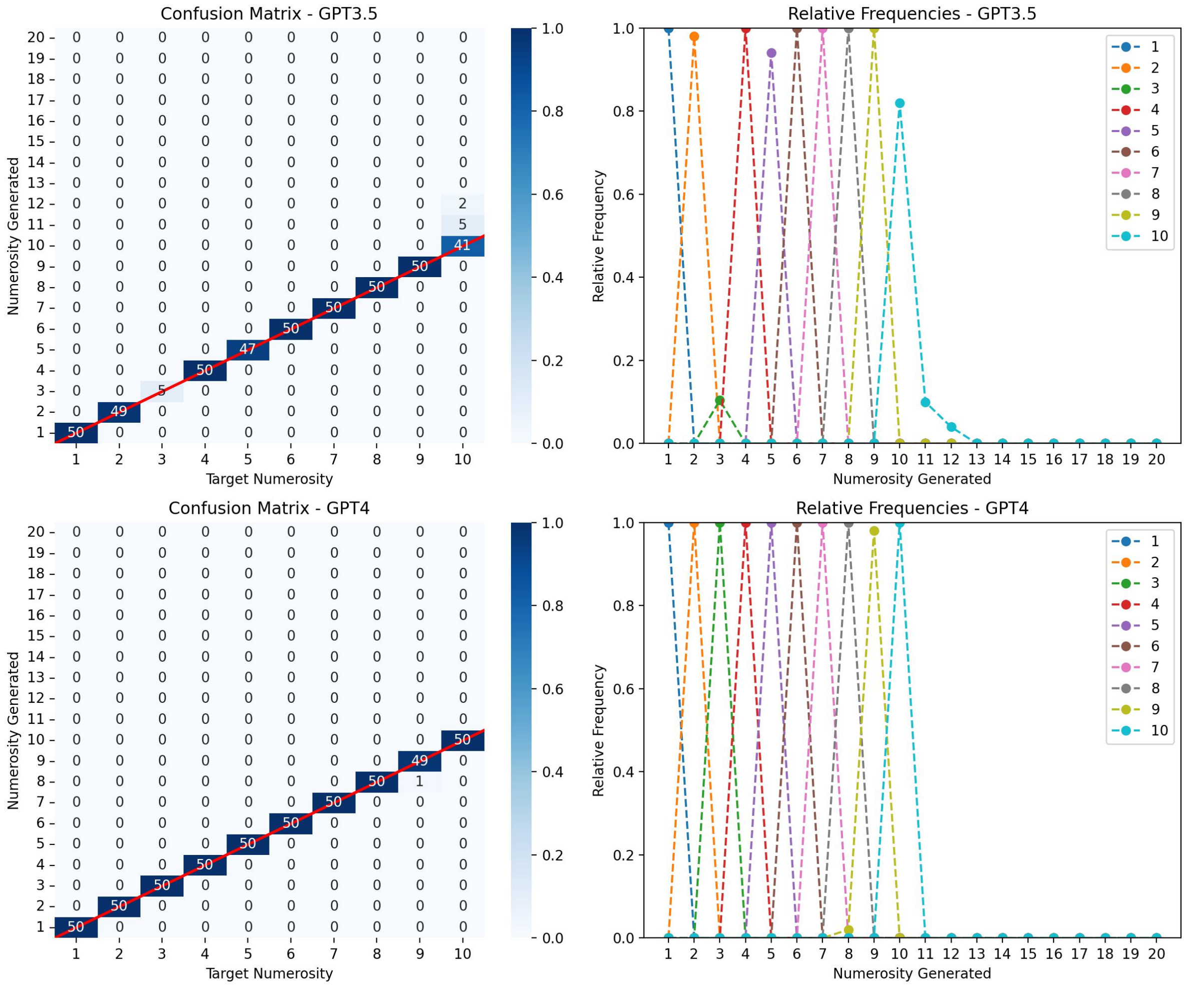

Figure 3.18: comparison of the confusion matrix and relative frequency between GPT-3.5 (top) and GPT-4 (bottom) where the condition is letter uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.
For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Word uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.974 and 1.0, respectively. Figure 3.19 shows the accuracies for each numerosity of both models.

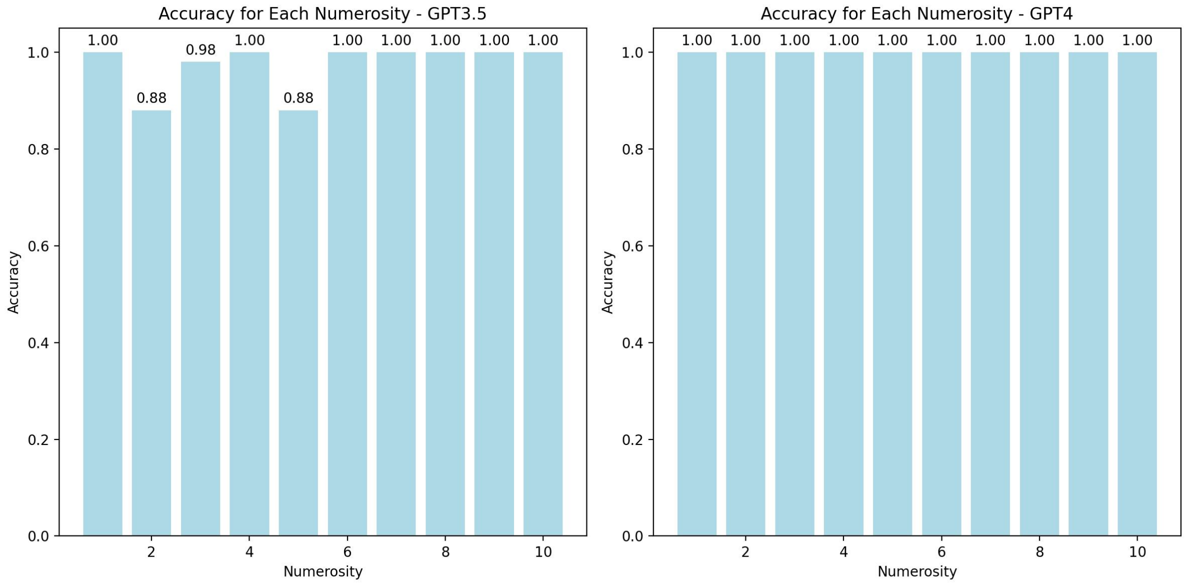

Figure 3.19: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is word uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix GPT-4 again counts perfectly. GPT-3.5 has some fluctuations and minor errors. Among these minor mistakes, it is the same situation as discussed in the non-uniform letter counting task that GPT-3.5 counts the repeated words as 1 (see Figure 3.20.

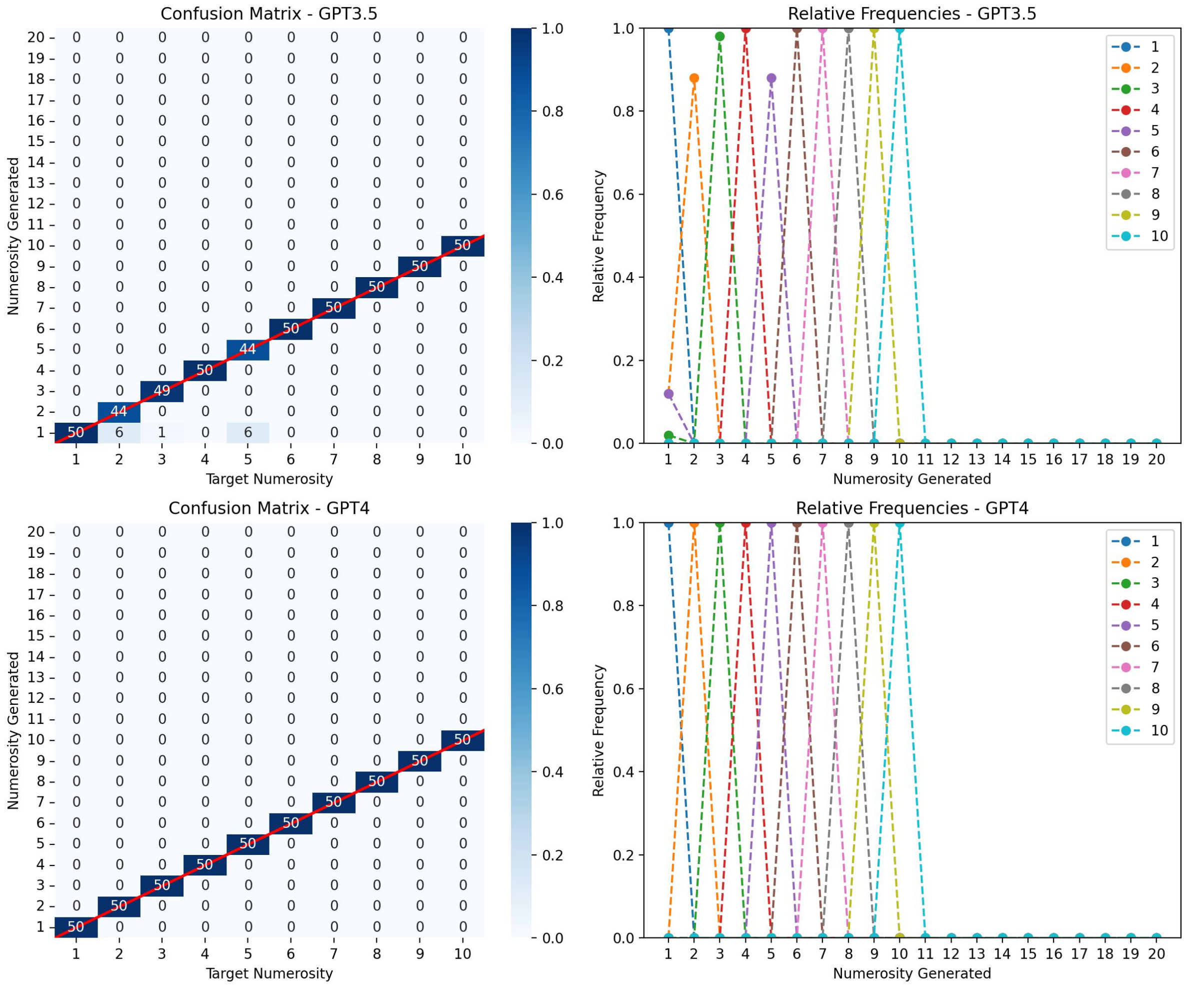

Figure 3.20: comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the condition is word uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Word non-uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.662 and 0.702, respectively. Figure 3.21 shows the accuracies for each numerosity of both models.

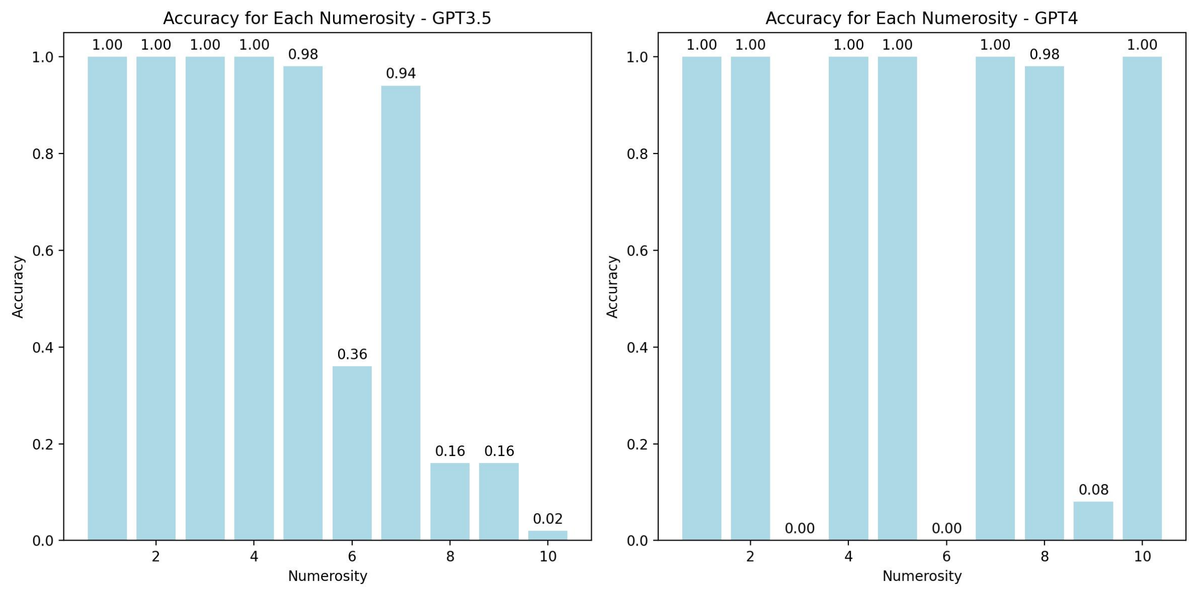

Figure 3.21: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is word non-uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix Compared with other conditions, both models count worse. It seems that GPT-3.5 is affected by the increasing numerosity (see Figure 3.22. However, for GPT-4, the reason why it fails to count on certain numerosities is uncertain. It could be the reason for tokenization, but to directly prove that it is due to tokenization is beyond the scope of this thesis.

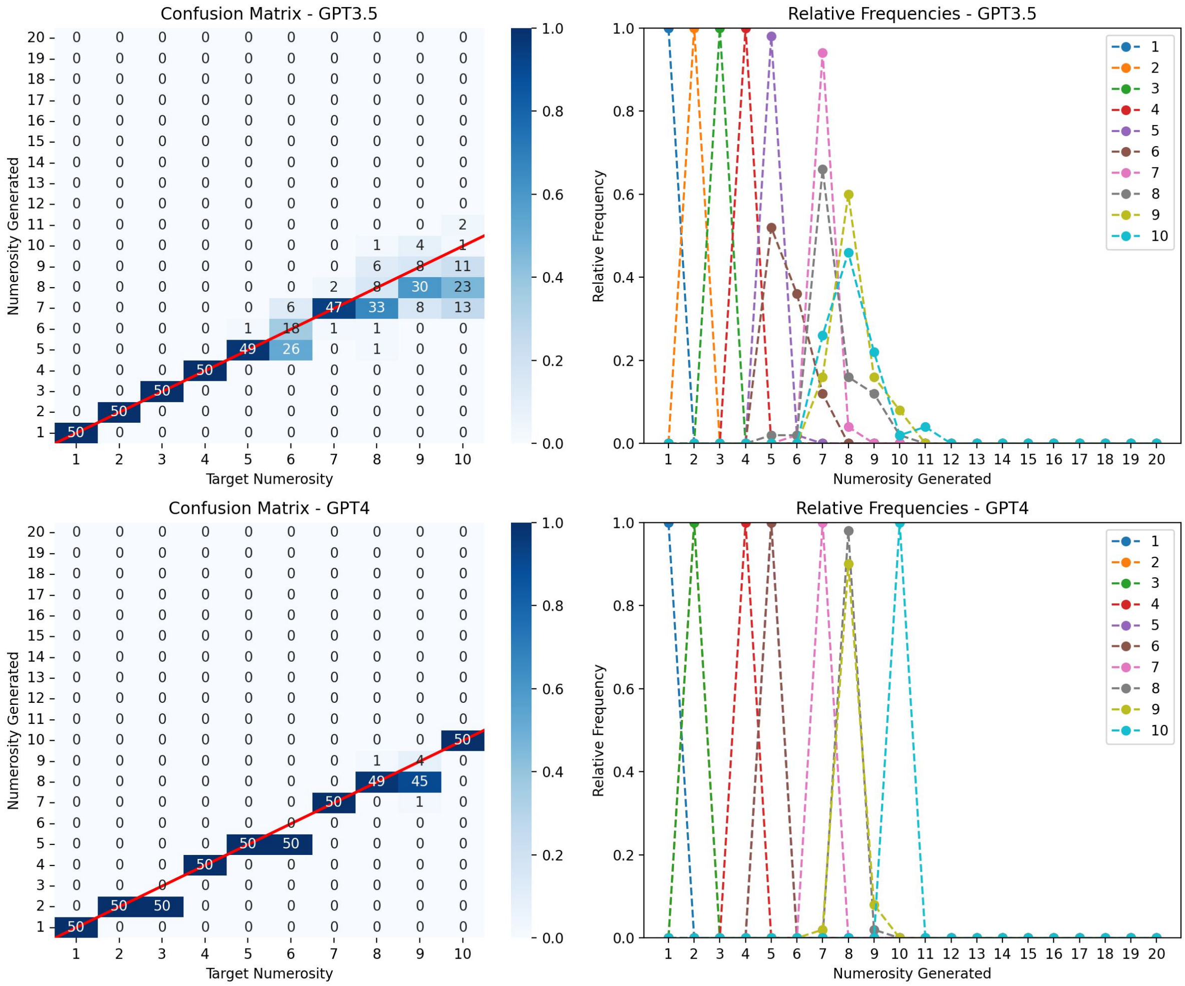

Figure 3.22: comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the condition is word non-uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

3.5 Numerosity Generation

3.5.1 Diffusion Models

Apple

Accuracy The overall accuracy for DALL·E and Stable Diffusion v2-1 is 0.456 and 0.302, respectively. Figure 3.23 shows the accuracies for each numerosity of both models.

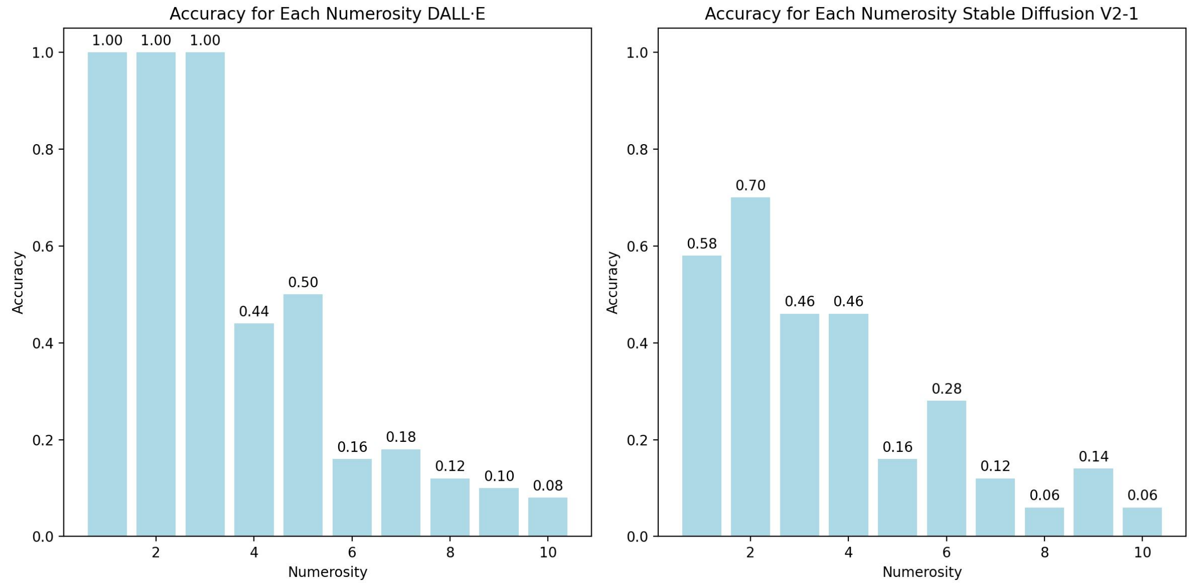

Figure 3.23: comparison between DALL·E (left) and SD2-1 (right) in terms of accuracy where the category is Apple. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix DALL·E generates the first 3 numerosities perfectly and until the numerosity reaches 5, DALL·E's errors are within acceptable limits. Once the numerosity goes beyond 5, DALL·E seems to generate the numerosities randomly. As for SD2-1, numerosities that are smaller than 5 are easy to generate, beyond that, it does not seem to understand how many apples to generate (see Figure 3.24.

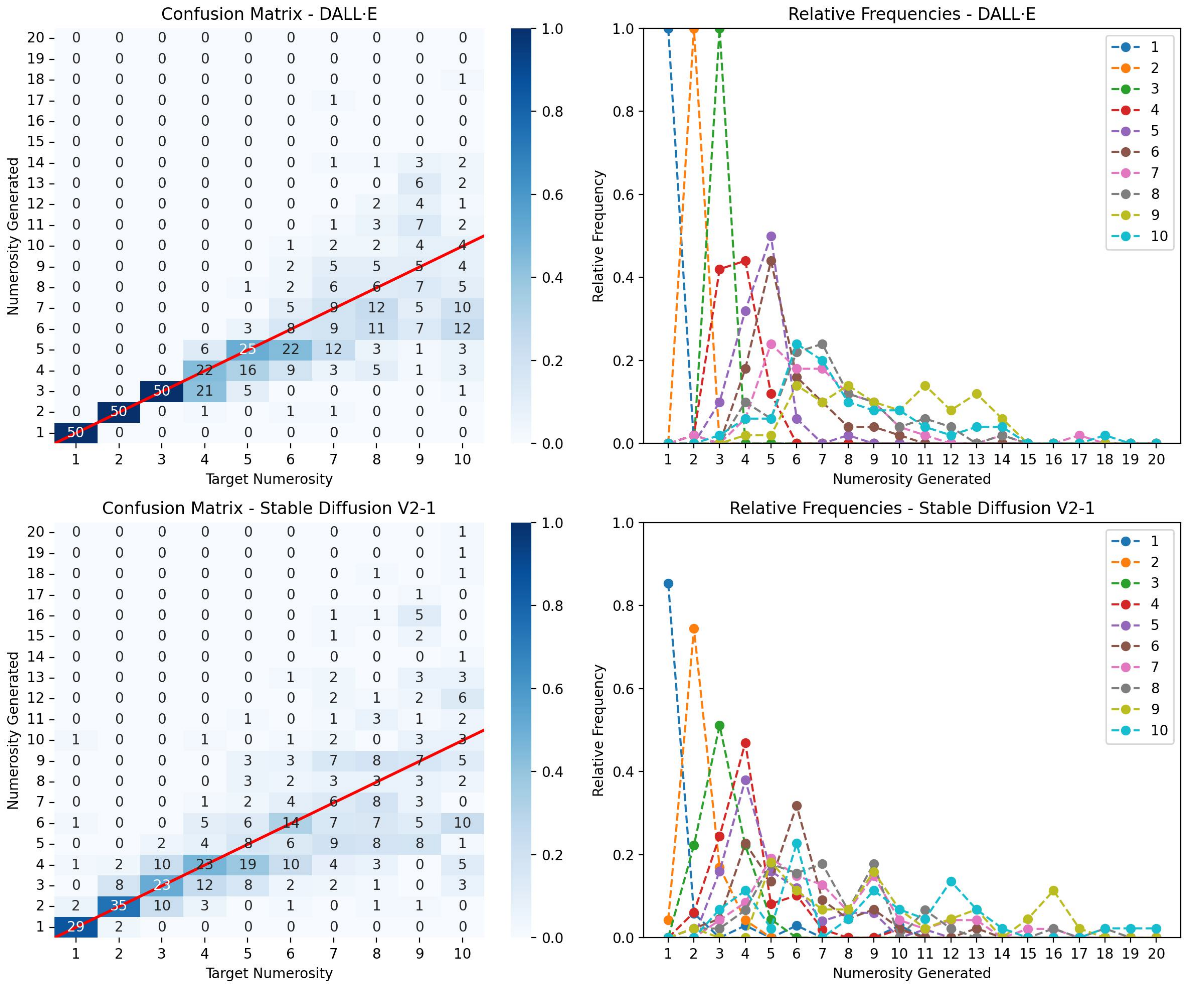

Figure 3.24: comparison of the confusion matrix and relative frequency between DALL·E (top) and SD2-1 (bottom) where the category is Apple. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Butterfly

Accuracy The overall accuracy for DALL·E and Stable Diffusion v2-1 is 0.410 and 0.312, respectively. Figure 3.25 shows the accuracies for each numerosity of both models.

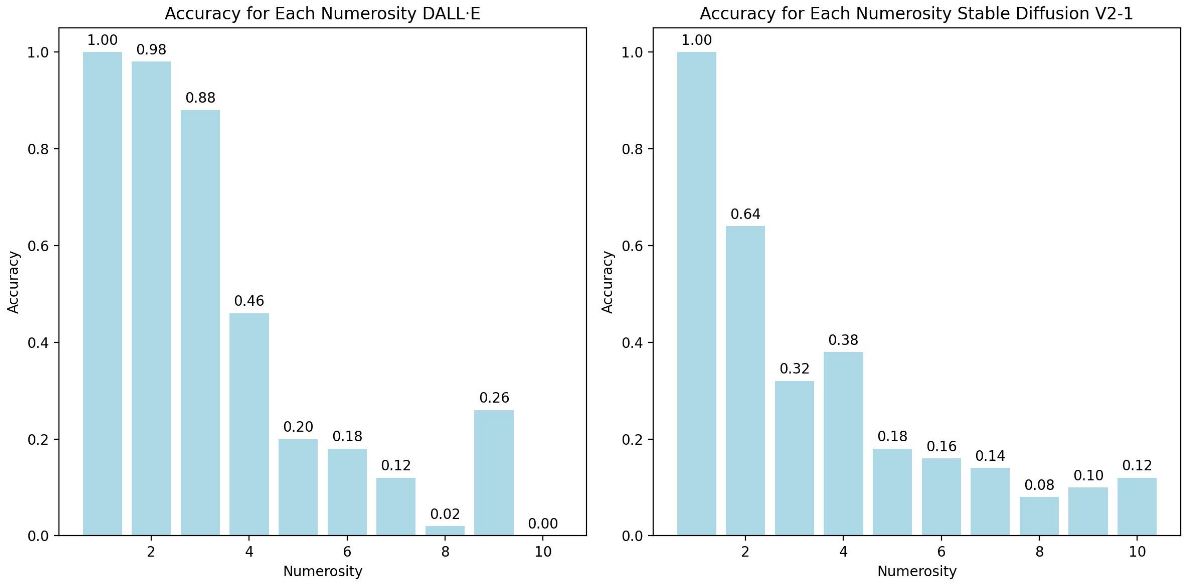

Figure 3.25: comparison between DALL·E (left) and SD2-1 (right) in terms of accuracy where the category is Butterfly. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix As shown in Figure 3.26 both models perfectly generate one butterfly. DALL·E maintains its performance until numerosity reaches 3, while SD2-1's performance plummets after 2. As numerosity gets larger and larger, both models are unable to give proper stably. The difference is that DALL·E tends to generate more butterflies than asked, while SD2-1 generates fewer butterflies than asked.

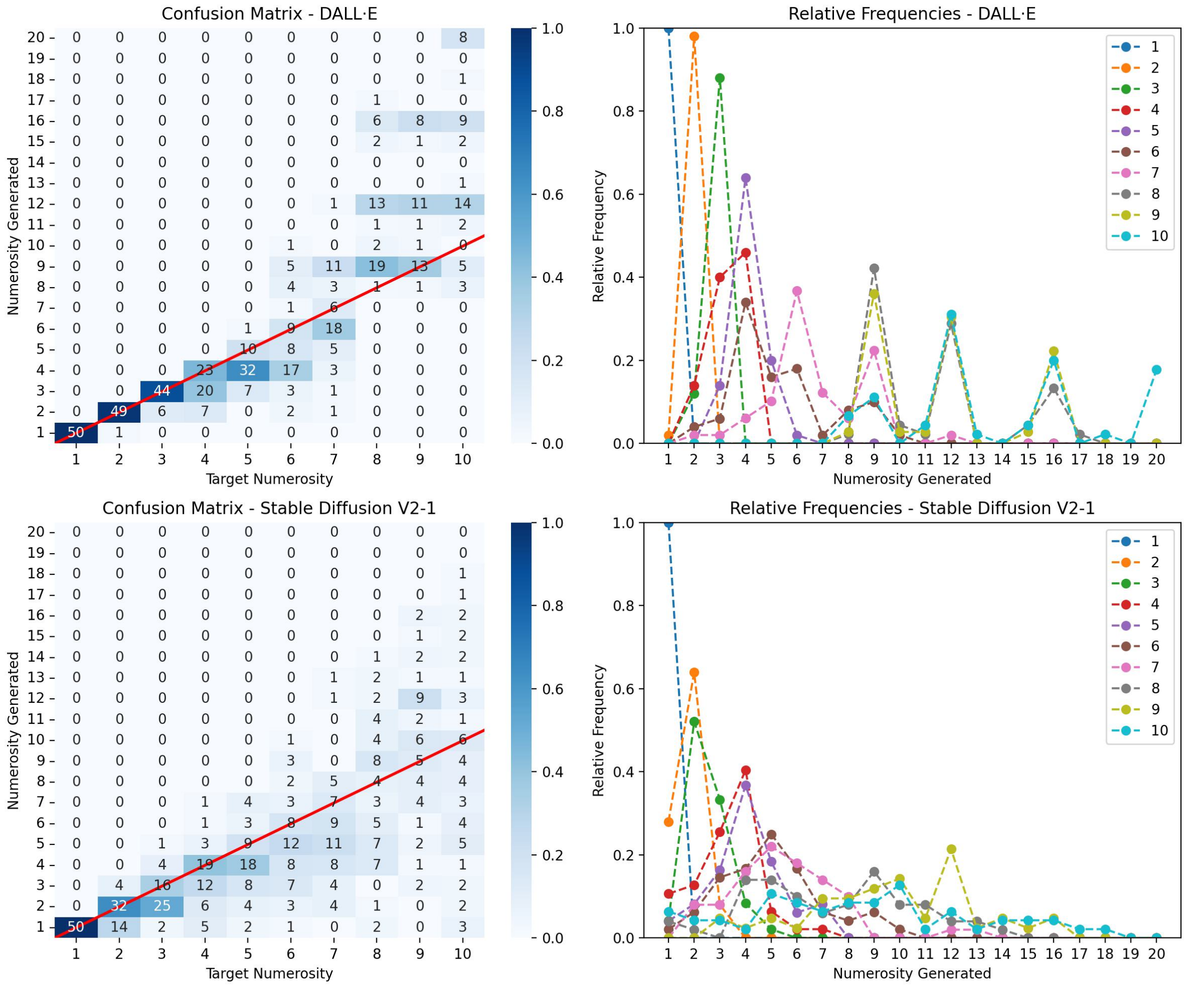

Figure 3.26: comparison of the confusion matrix and relative frequency between DALL·E (top) and SD2-1 (bottom) where the category is Butterfly. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Dot

Accuracy The overall accuracy for DALL·E and Stable Diffusion v2-1 is 0.130 and 0.038, respectively. Figure **??** shows the accuracies for each numerosity of both models.

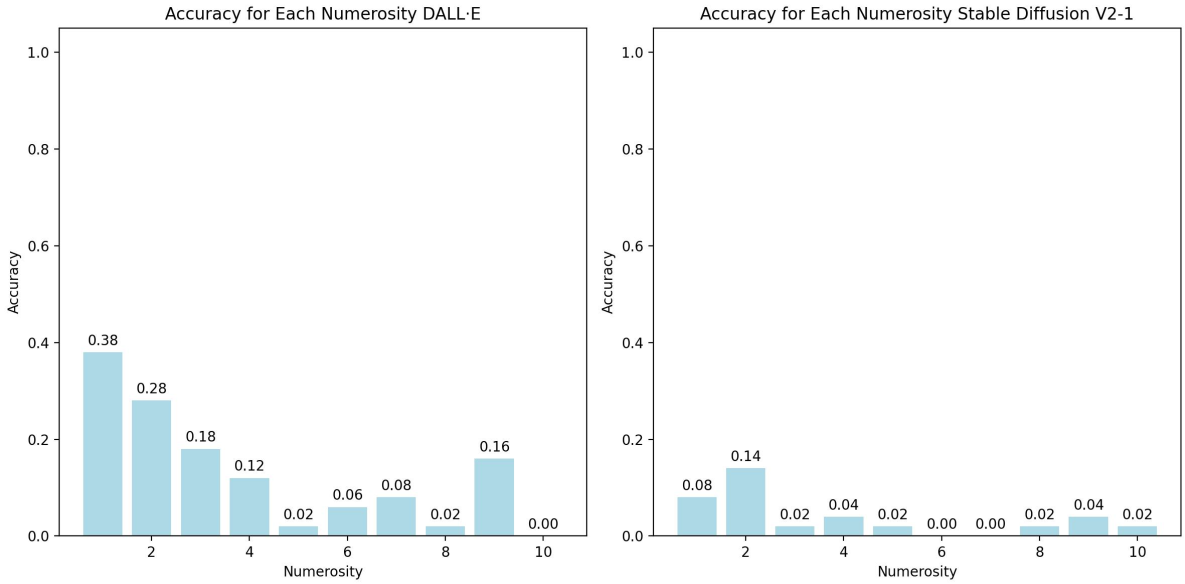

Figure 3.27: comparison between DALL·E (left) and SD2-1 (right) in terms of accuracy where the category is Dot. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix For both models, it is too challenging to generate the required number of points. Even 1 dot is not easy for both models. In general, DALL·E still outperforms SD2-1 (see Figure 3.28).

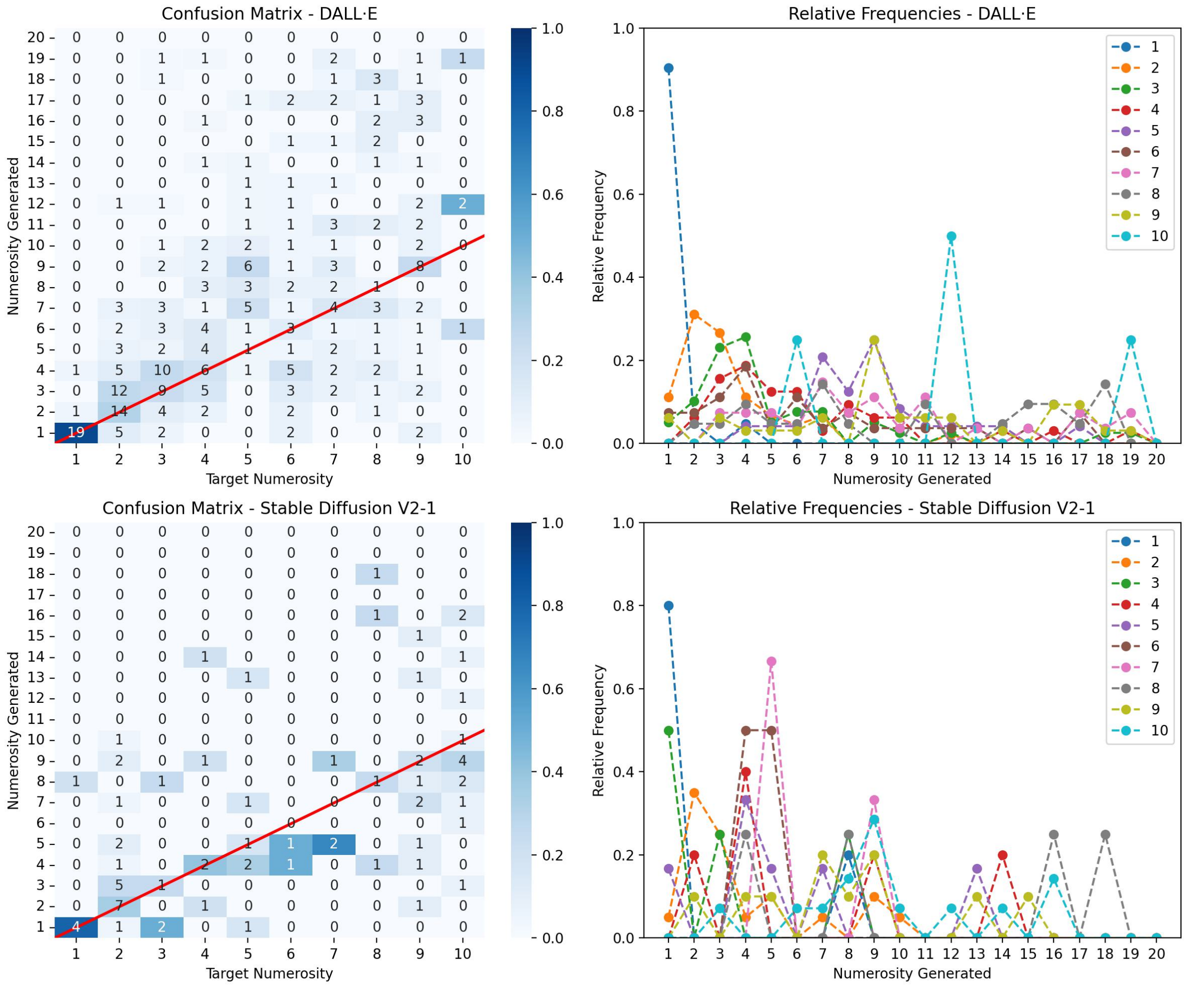

Figure 3.28: comparison of the confusion matrix and relative frequency between DALL·E (top) and SD2-1 (bottom) where the category is Dot. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Flashccard

Accuracy The overall accuracy for DALL·E and Stable Diffusion v2-1 is 0.404 and 0.218, respectively. Figure 3.29 shows the accuracies for each numerosity of both models.

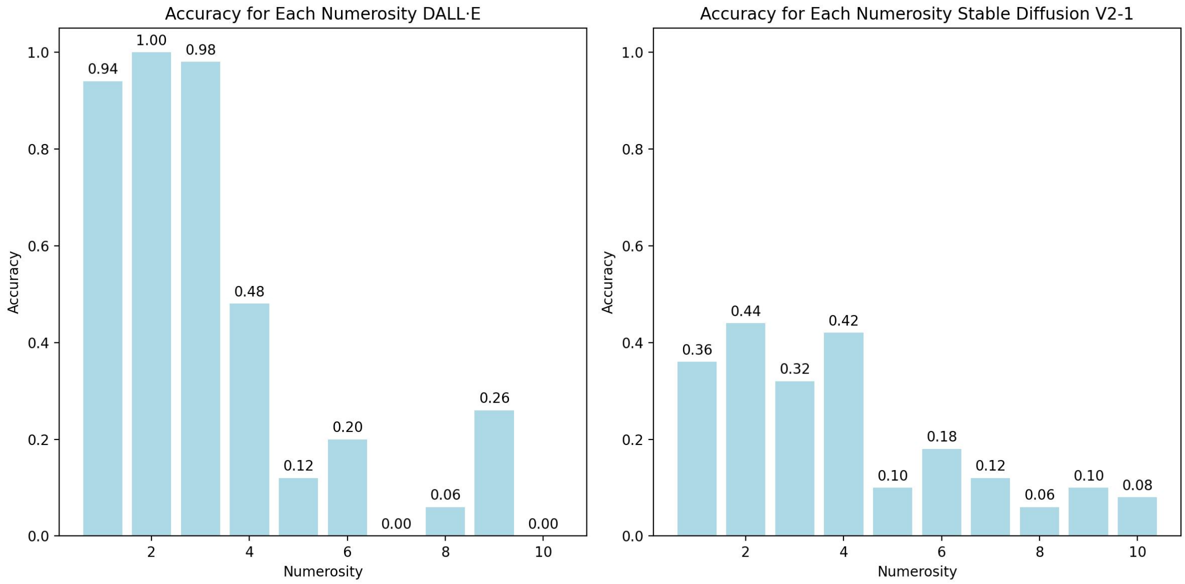

Figure 3.29: comparison between DALL·E (left) and SD2-1 (right) in terms of accuracy where the category is Flashcard. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix For the first numerosities, DALL·E keeps a high accuracy. As the numerosity gets larger, DALL·E tends to generate more flashcards and SD tends to generate fewer flashcards (see Figure 3.30). This trend is also found when the category is Butterfly.

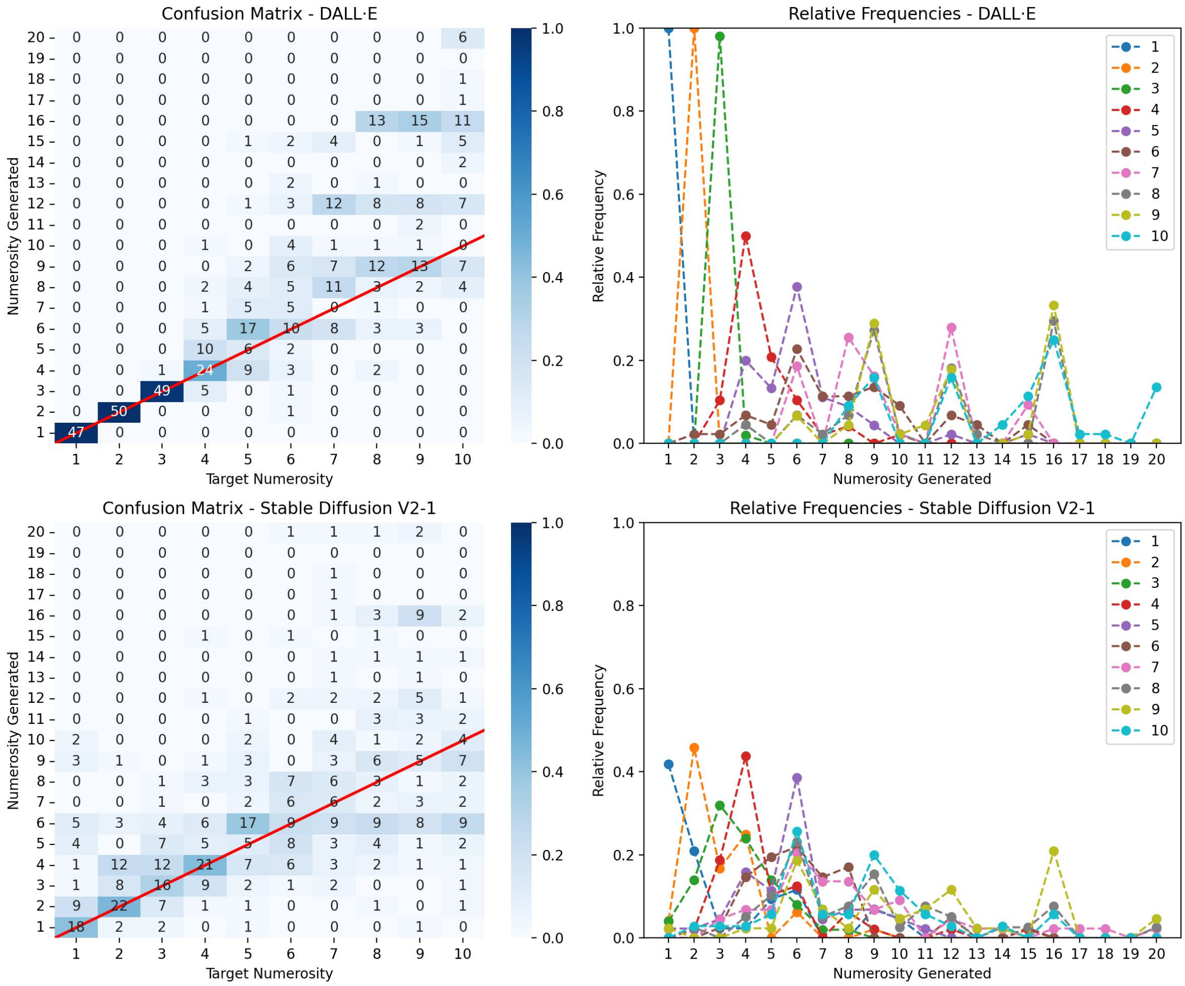

Figure 3.30: comparison of the confusion matrix and relative frequency between DALL·E (top) and SD2-1 (bottom) where the category is Flashcard. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

People

Accuracy The overall accuracy for DALL·E and Stable Diffusion v2-1 is 0.382 and 0.180, respectively. Figure 3.31 shows the accuracies for each numerosity of both models.

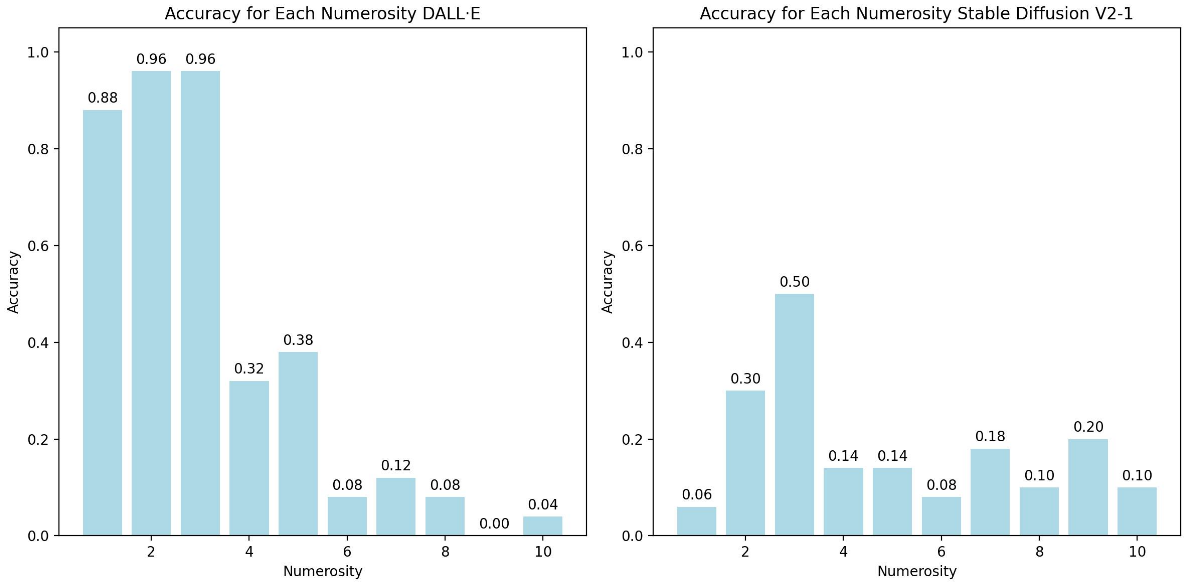

Figure 3.31: comparison between DALL·E (left) and SD2-1 (right) in terms of accuracy where the category is People. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix Once again, DALL·E generates the first 3 numerosities almost perfectly(see Figure 3.32). SD2-1 generates a random number of people for all numerosites except for 2 and 3, where the accuracy is clearly higher than other numerosites. Taking a look at the confusion matrix, DALL·E has its tendency to generate more people than asked. However, for SD2-1, the generated numerosities are distributed throughout the matrix. It could be the reason that SD2-1 does not focus on numerosity but rather on the singular and plural of a noun.

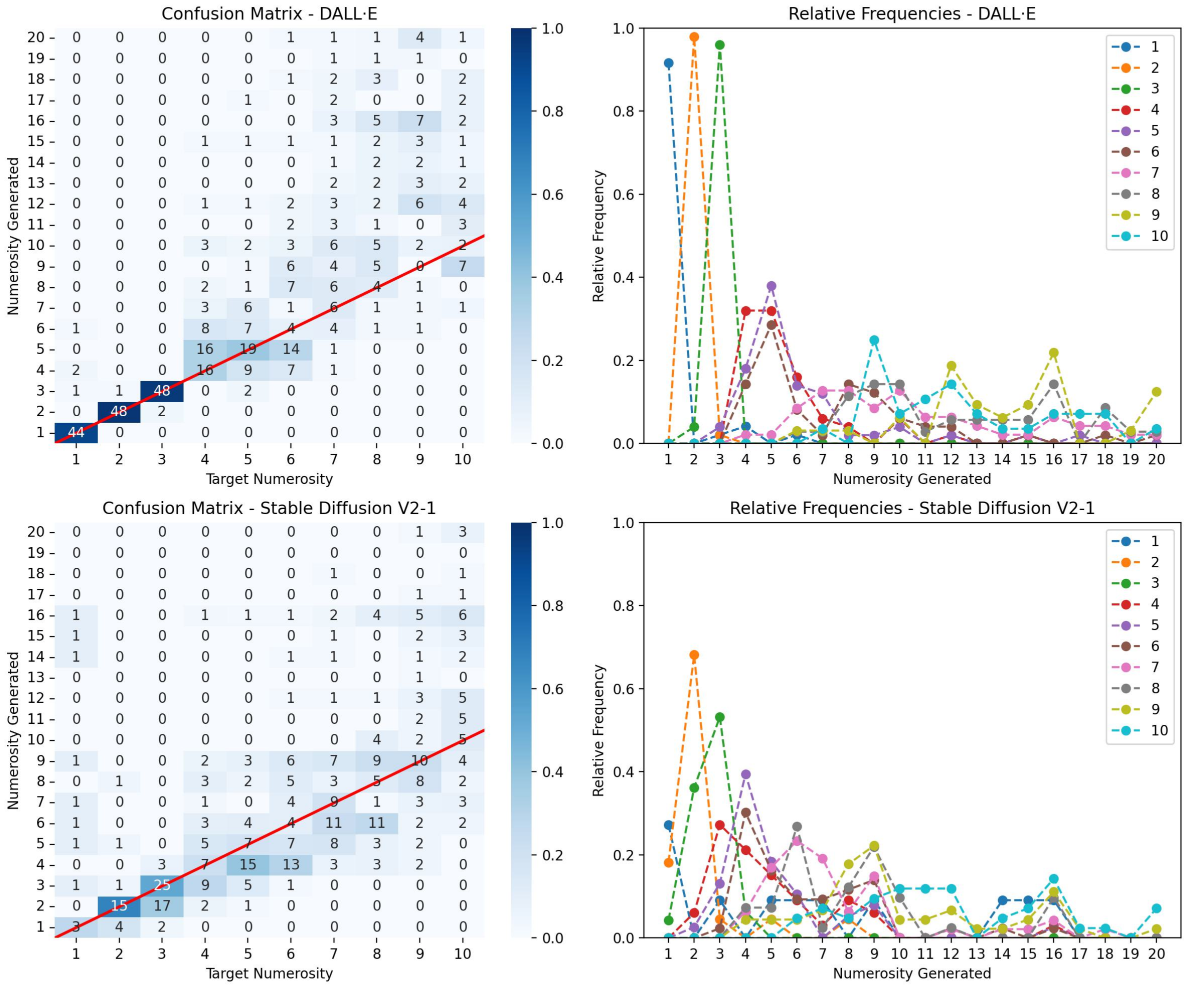

Figure 3.32: comparison of the confusion matrix and relative frequency between DALL·E (top) and SD2-1 (bottom) where the category is People. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance. For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

3.5.2 GPT Models

Letter non-uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.98 and 1.0, respectively. Figure 3.33 shows the accuracies for each numerosity of both models.

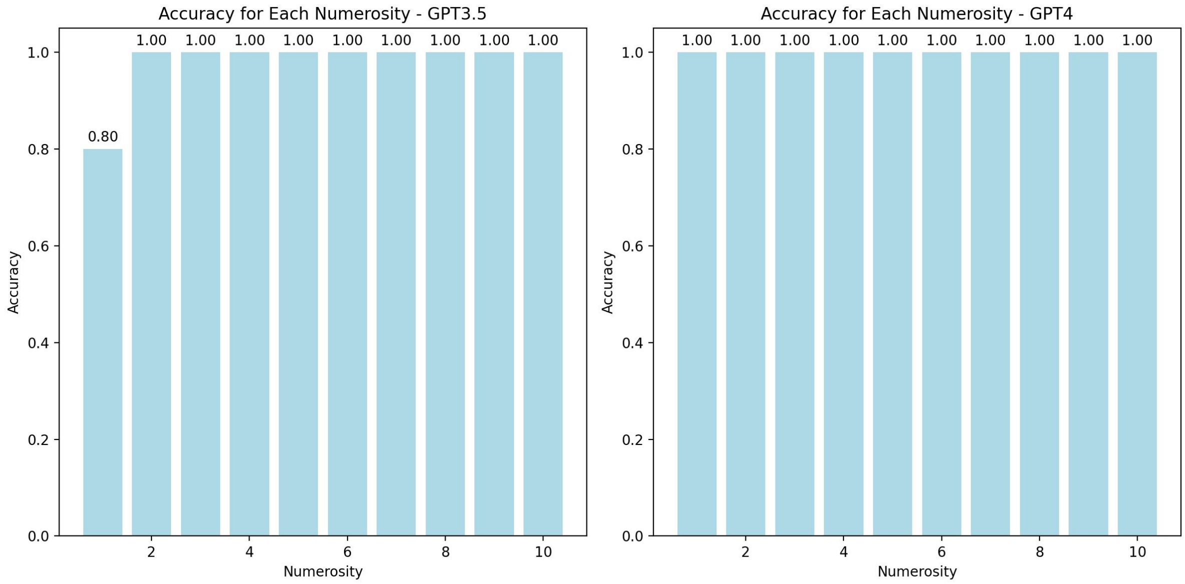

Figure 3.33: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is letter non-uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix GPT-4 generates the non-uniform strings without any mistakes. GPT-3.5 makes mistakes on numerosity 1 (see Figure 3.34). The string to be generated is ”umb” and 20% of the responses repeat the string twice.

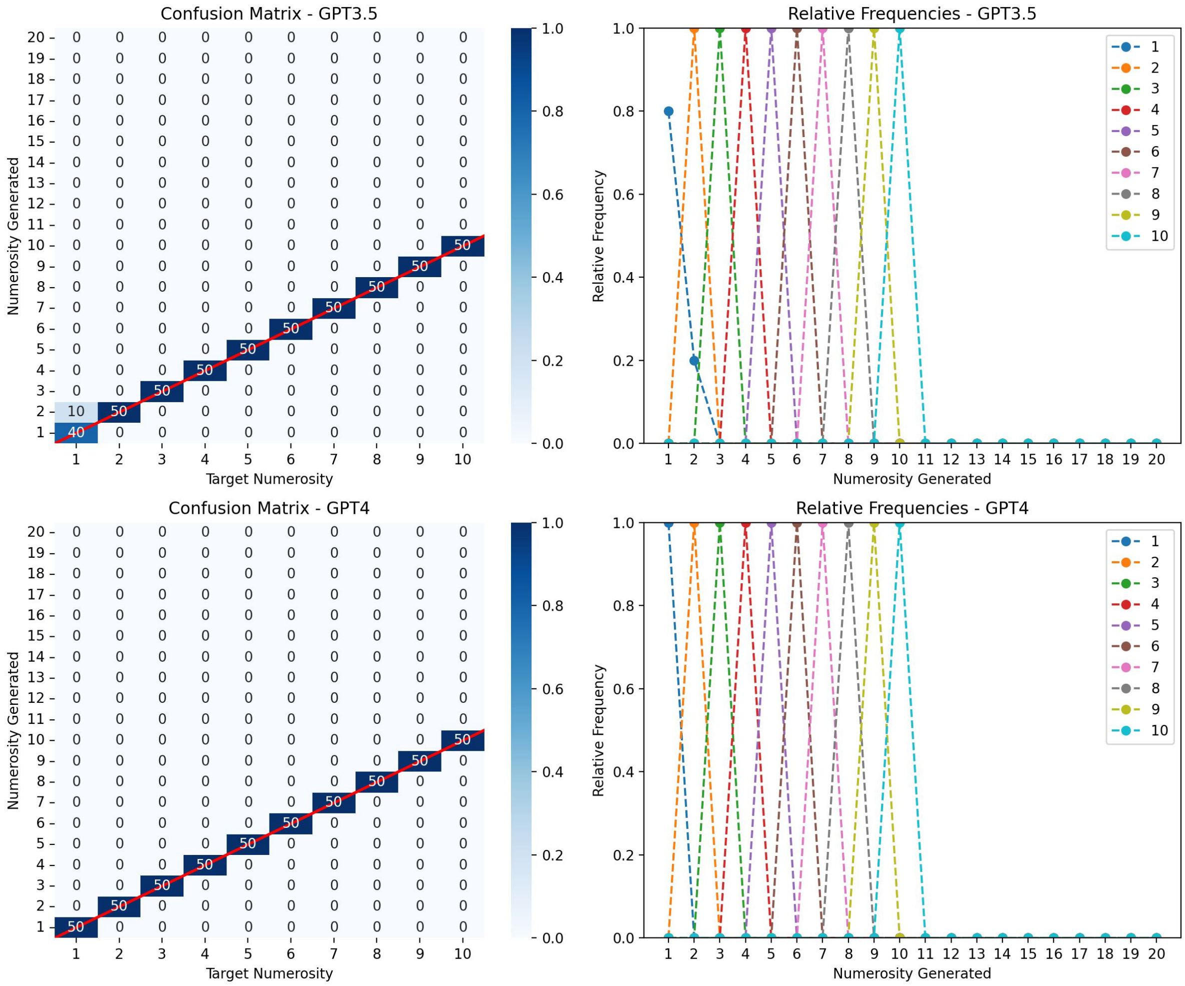

Figure 3.34: comparison of the confusion matrix and relative frequency between GPT-3.5 (top) and GPT-4 (bottom) where the condition is letter non-uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Letter uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.994 and 1.0, respectively. Figure 3.35 shows the accuracies for each numerosity of both models.

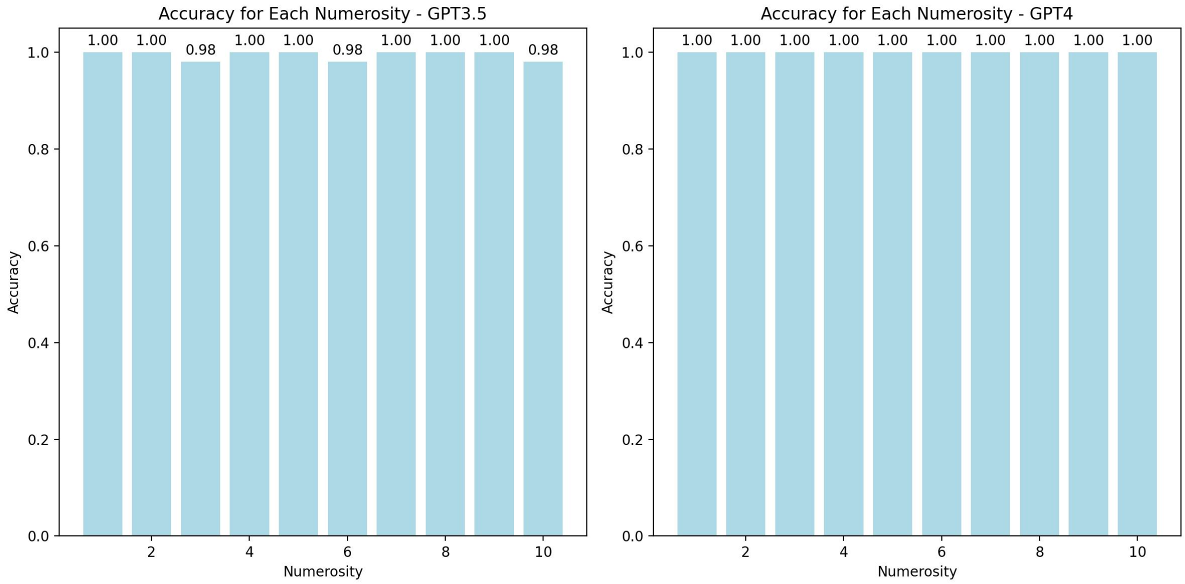

Figure 3.35: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is letter uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix Both models deliver well-enough performance. Some minor fluctuations remain in the responses of GPT-3.5 (see Figure 3.36.

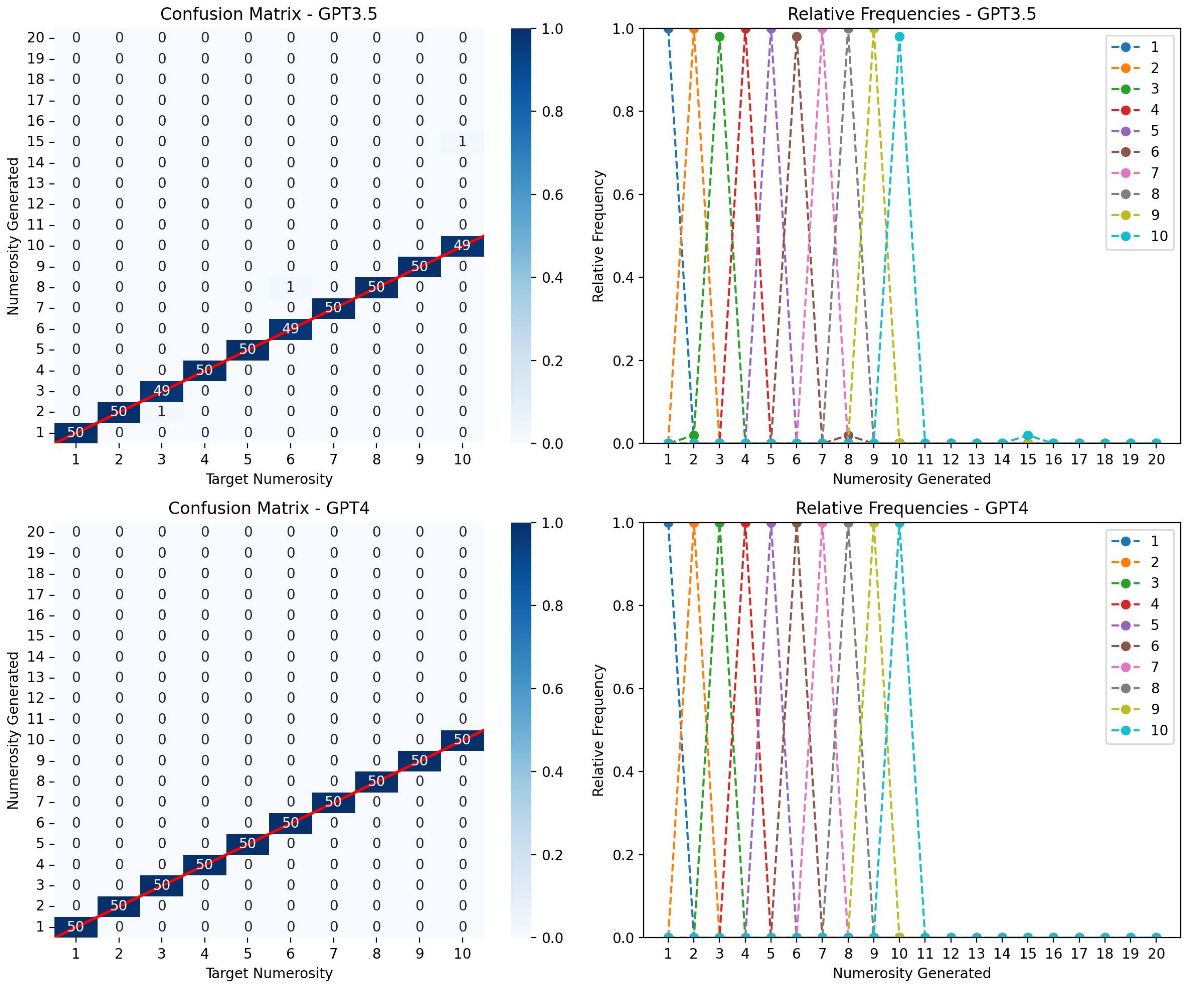

Figure 3.36: comparison of the confusion matrix and relative frequency between GPT-3.5 (top) and GPT-4 (bottom) where the condition is letter uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Word uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.998 and 1.0, respectively. Figure 3.37 shows the accuracies for each numerosity of both models.

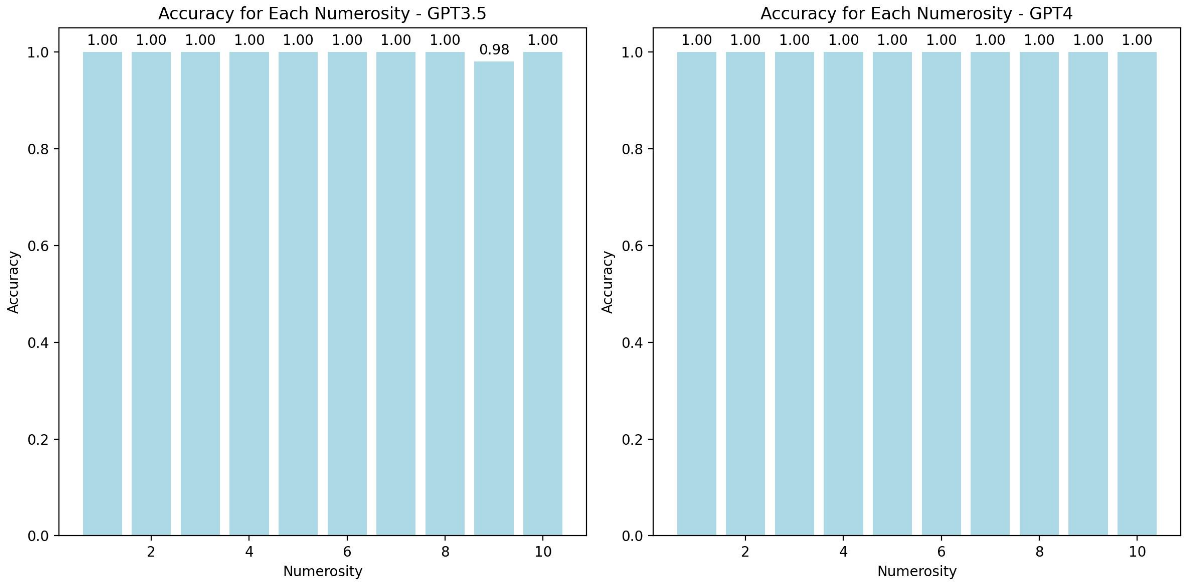

Figure 3.37: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is word uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix Both models deliver good performance. Some minor fluctuations remain in the responses of GPT-3.5 (see Fiugre 3.38.

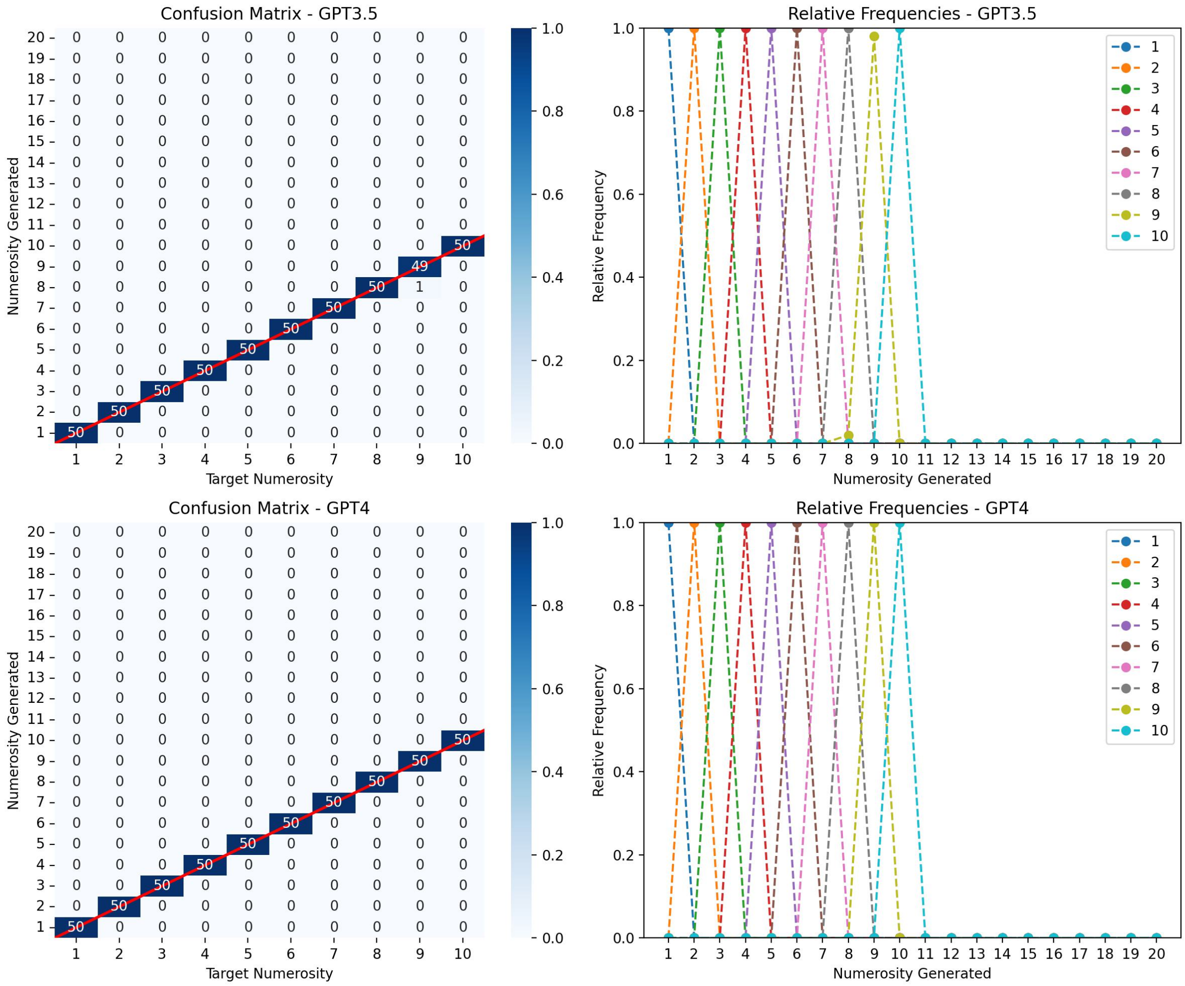

Figure 3.38: comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the condition is word uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.

For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

Word non-uniform

Accuracy The overall accuracy for GPT-3.5 and GPT-4 is 0.640 and 0.998, respectively. Figure 3.39 shows the accuracies for each numerosity of both models.

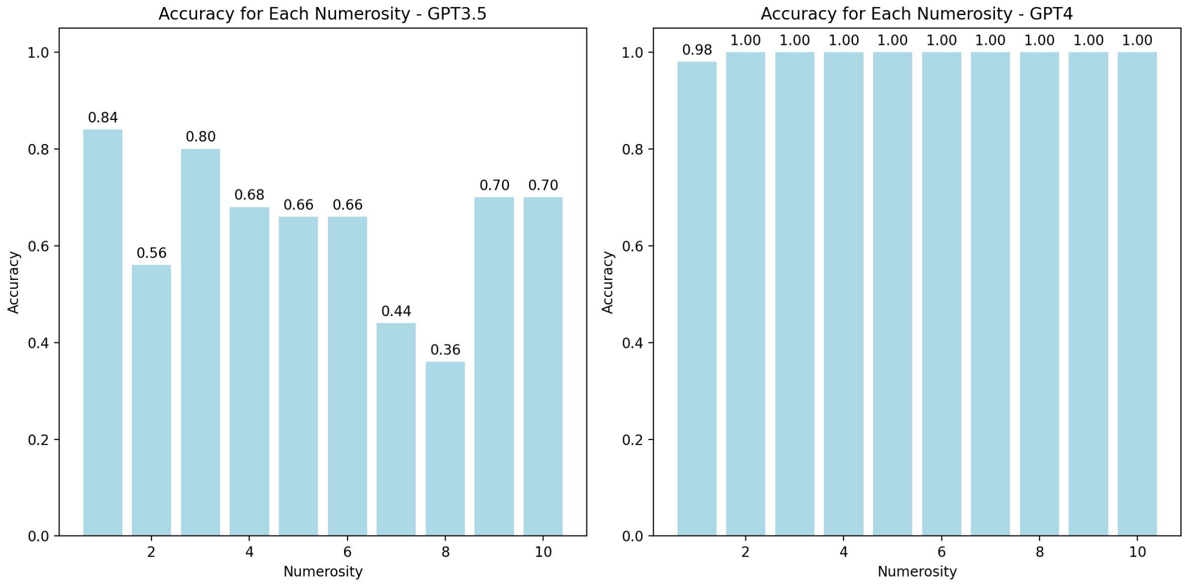

Figure 3.39: comparison between GPT-3.5 (left) and GPT-4 (right) in terms of accuracy where the condition is word non-uniform. Each bar represents a numerosity, the y-axis stands for accuracy.

Relative Frequency and Confusion Matrix GPT-4 once again shows powerful performance on this task. GPT-3.5 is clearly affected by the increasing numerosity (see Figure 3.40).

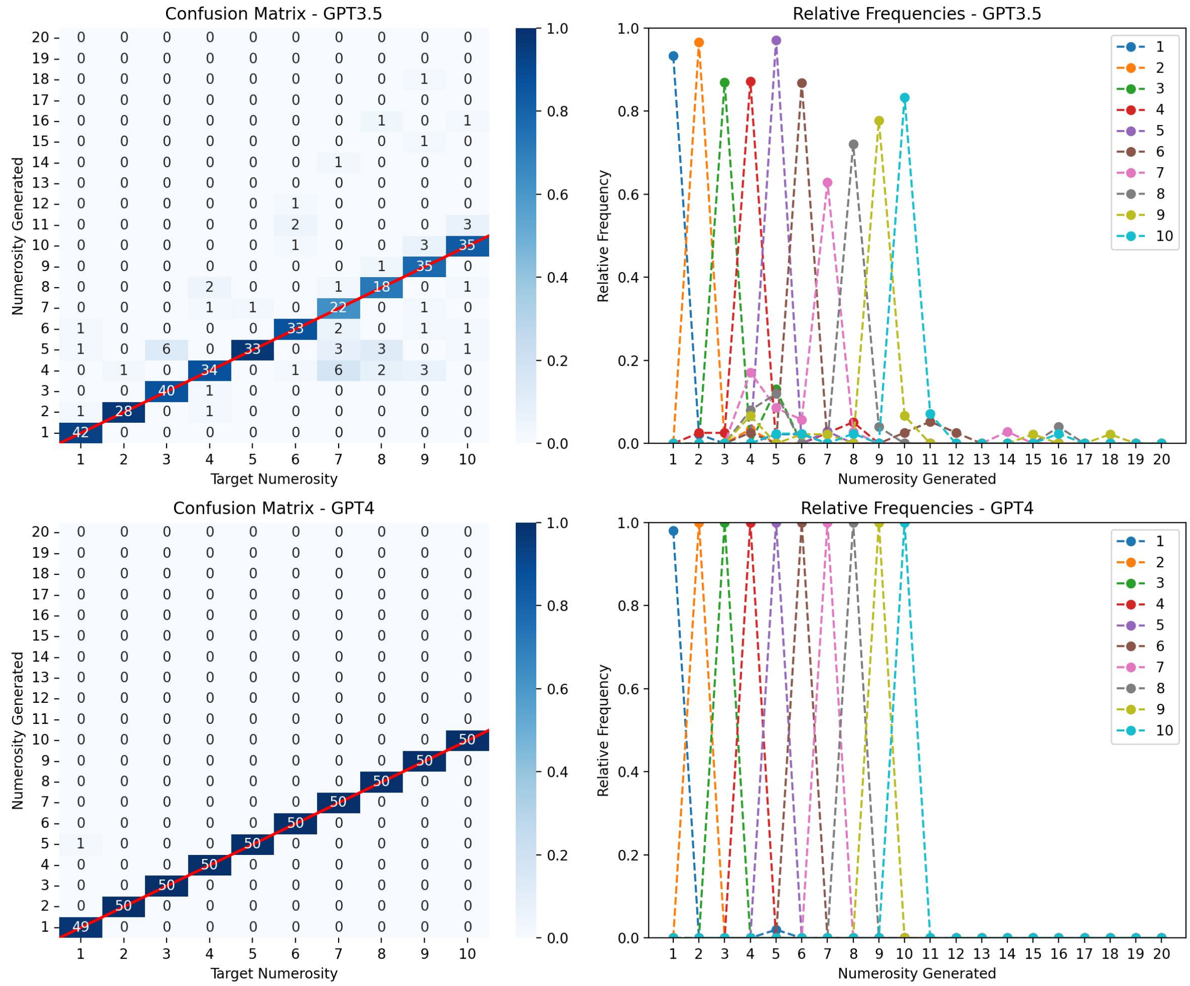

Figure 3.40: comparison of the confusion matrix and relative frequency between ViLT (top) and BLIP (bottom) where the condition is word non-uniform. The y-axis of the confusion matrix represents the numerosity generated by the models and the x-axis represents the target numerosity that should be generated. The closer the distribution is to the red diagonal, the better the performance.
For the relative frequency plot, the y-axis is the relative frequency of each generated numerosity; each dashed line represents the 10 numerosities from 1 to 10.

3.6 Multimodal (visual-language) tasks

3.6.1 Geometrical Concepts

2 ways of measuring the performance of the models are applied here.

Score:

1 point: if the model answers or draws exactly what the prompt asks.

0.5 point: if the model answers or draws the subclass or superclass of what the prompt asks.

0 point: if none of the above

Correct Trails:

Only if the model answers or draws exactly what the prompt asks.

ViLT and BLIP 11 images created with Python matplotlib, Polygon, and Numpy libraries are tested. These 11 geometrical plane shapes are tested with the prompt:

"What is this geometrical shape?"

The reasoning process is completed with a single precision. The total score for both models is 5.5 and the correct trials are 3. It shows that both models tend to tell the general superclass of the geometrical shape when it comes to triangles and to recognize quadrangles as squares. When faced with a trapezoid, both models cannot perform the reasoning process correctly, that is, to count the number of sides of the polygon, but rather to give an approximate estimate of the shape of the polygon. Since triangles and trapezoids are similar on pixel levels, there is no doubt that both models failed to recognize it correctly. A detailed discussion of why both models fail can be found in the next chapter.

Table 3.2: Detailed ViLT Responses of Geometrical Shape Recognition

Image	**Results**	**Score**	**Correct Trails**
circle	circle	1	1
hexagon	diamond	0	0
isosceles_trapezoid	triangle	0	0
isosceles_triangle	triangle	0.5	0
oval	circle	0.5	0
rectangle	square	0	0
right_triangle	triangle	0.5	0
square	square	1	1
trapezoid	triangle	0	0
triangle	triangle	1	1
equilateral triangle	triangle	0.5	0

Table 3.3: Detailed BLIP Responses of Geometrical Shape Recognition

Image	Results	Scores	Correct Trails
circle	circle	1	1
hexagon	octagon	0	0
isosceles_trapezoid	triangle	0	0
isosceles_triangle	triangle	0.5	0
oval	circle	0.5	0
rectangle	square	0	0
right_triangle	triangle	0.5	0
square	square	1	1
trapezoid	triangle	0	0
triangle	triangle	1	1
equilateral triangle	triangle	0.5	0

Instruction in the Prompts It is possible that both models tend to make mistakes, so that when faced with difficult tasks, both models choose the answer that is not wrong, but also not correct. To find out whether this is some kind of trade-off that was learned by the models during training or it is purely because the models cannot recognize different types of geometrical shapes, an instruction is added to the prompt:

"what is this geometrical shape? answer in detail. e.g. if a triangle's 3 sides have the same length, then answer with an equilateral triangle instead of triangle"

By adding this instruction, the models should be forced to be specific in geometrical shapes. However, that is not the case, instead, the models perform worse in this situation. Both models end with a total score of 3 and correct trials of 2. It is worth noting that both models tried to follow the instructions but did not follow them correctly.

Counting the Number of Sides of Polygon In the previous tests, both models did not recognize some geometrical shapes, especially polygons. One possible reason could be that both models cannot count the numbers of the sides of a polygon. Polygons with sides ranging from 3 to 12 are tested.

The results show that both models fail, where ViLtT only does 2 trails out of 10 correctly

Table 3.4: Instruction in Prompt, responses from ViLT

Image	Results	Scores	Correct Trails
circle	triangle	0	0
hexagon	hexagon	0	0
isosceles_trapezoid	triangle	0	0
isosceles_triangle	triangle	0.5	0
oval	triangle	0	0
rectangle	triangle	0	0
right_triangle	triangle	0.5	0
square	triangle	0	0
trapezoid	triangle	0	0
triangle	triangle	1	1
equilateral triangle	triangle	1	1

Table 3.5: Instruction in Prompt, responses from BLIP

Image	Results	Scores	Correct Trails
circle	triangle	0	0
hexagon	diamond	0	0
isosceles_trapezoid	triangle	0	0
isosceles_triangle	triangle	0.5	0
oval	triangle	0	0
rectangle	triangle	0	0
right_triangle	triangle	0.5	0
square	triangle	0	0
trapezoid	triangle	0	0
triangle	triangle	1	1
equilateral triangle	triangle	1	1

and where BLIP shows a strong preference of the number 8 and only does 1 trail correctly. Both models cannot count the number of sides of a polygon.

Table 3.6: Detailed Responses of Counting Sides of a Polygon

Image	**Results_ViLT**	**Results_BLIP**
10_sides_polygon	8	8
11_sides_polygon	8	8
12_sides_polygon	8	8
3_sides_polygon	4	8
4_sides_polygon	4	8
5_sides_polygon	4	8
6_sides_polygon	8	8
7_sides_polygon	8	8
8_sides_polygon	8	8
9_sides_polygon	8	8

Diffusion Models 18 trials are instructed for 4 diffusion models separately. None of the models performs well. The commercial models (i.e. DALL·E & Stable Diffusion V2-2_XL) tend to draw a more detailed and aesthetically better image, but the overall understanding of basic plane geometry is not improved (see Figure 3.41

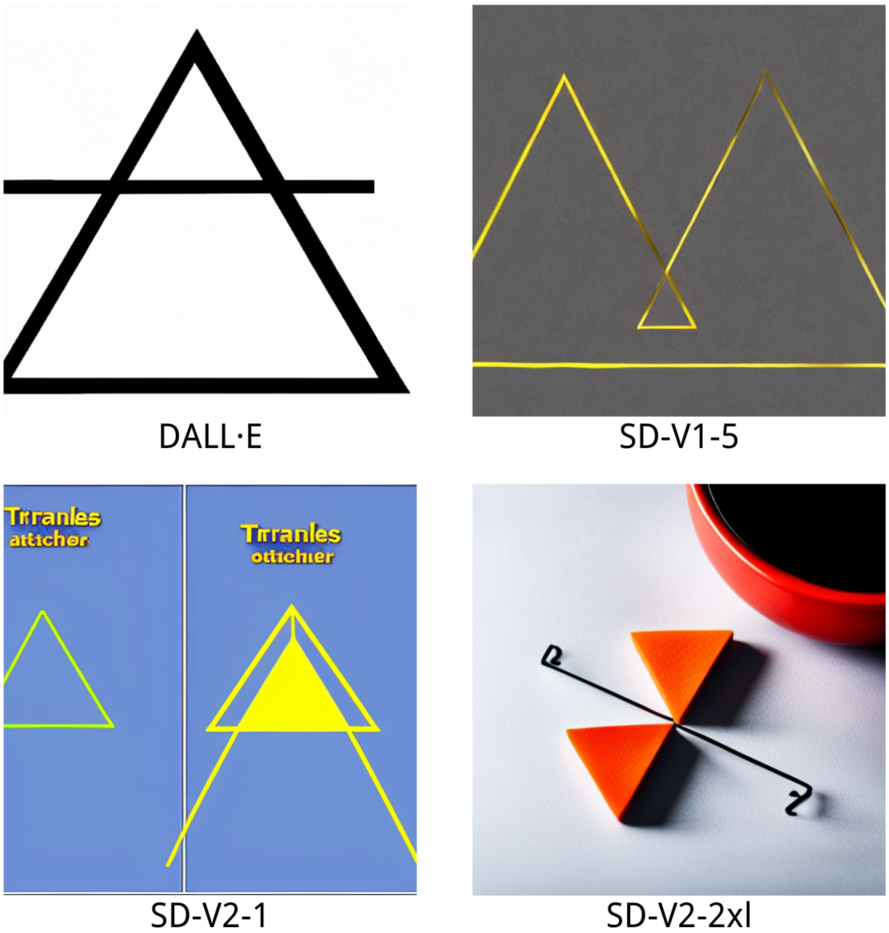

Figure 3.41: An example of how four models respond to the prompt: ”2 triangles that are congruent to each other”

Prompts:

”a line A through dot B so that line A represents the closest distance between the dot and the line ”,

”2 lines that are parallel to each other”,

”2 lines that are perpendicular to each other”,

”a line that is perpendicular to the line”,

”a right triangle”,

”a isosceles triangle”,

”a equilateral triangle”,

”the midpoint of a triangle”,

”the median bisector of a triangle”,

”the height bisector of a triangle”,

”the angle bisector of a triangle”,

”2 triangles that are congruent to each other”,

”a rectangle”,

”a square, plane geometric, math

”a rhomboid”,

”a trapezoid”,

”a isosceles trapezoid”,

”a circle, plane geometric, math”

Table 3.7: Results of Geometrical Concept by Diffusion Models

Models	**Score**	**Correct Trails**
DALL·E	3	3
SD V1-5	1	1
SD V2-1	5	4
SD V2-2_XL	4	3

3.6.2 Numerosity Comparison Vision

The numerosity comparison task is carried out by ViLT and BLIP. Accuracy with respect to each target number is calculated. The Weber fraction is calculated in 2 ways. The estimated one is by fitting the experimental results into either a log-Gaussian model or a scalar variability model denoted by

Log-Gaussian:

$$\omega_e = \sqrt{\log\left(1 + \left(\frac{1}{\text{Accuracy}}\right)^2\right)}$$

Scalar Variability:

$$\text{sd} = \frac{\text{target}}{\text{Accuracy}}$$

$$\omega_e = \frac{\text{sd}}{\text{target}}$$

The absolute one is calculated trail by trail (only correct responses are taken into consideration) and averaged across each target number, denoted by

$$\omega_a = \frac{|\text{target} - \text{reference}|}{\text{reference}}$$

ViLT ViLT performs poorly on this task. Once the target number goes above one, ViLT begin to guess and the final accuracy stays at chance level.

Table 3.8: ViLT results in numerosity comparison task when size and magnitude is congruent

Target_num	**Accuracy**	**Weber_Absolute**	**Weber_Estimate**
1	0.5	3	1.055
2	0.5	2.091	1.269
3	0.5	1.611	1.269
4	0.5	1.317	1.269
5	0.5	1.121	1.201
6	0.5	0.983	1.335
7	0.5	0.882	1.269
8	0.55	0.802	1.269
9	0.5	0.74	1.269
10	0.5	0.689	1.269
11	0.5	0.646	1.269
12	0.5	0.608	1.269

Table 3.9: ViLT results in numerosity comparison task when size and magnitude are incongruent

Target_num	Accuracy	Weber_Absolute	Weber_Estimate
1	0.5	3	1.055
2	0.5	2.091	1.269
3	0.5	1.611	1.269
4	0.5	1.317	1.269
5	0.5	1.121	1.201
6	0.5	0.983	1.335
7	0.5	0.882	1.269
8	0.55	0.802	1.269
9	0.5	0.74	1.269
10	0.5	0.689	1.269
11	0.5	0.646	1.269
12	0.5	0.608	1.269

It is worth notifying that the congruency of magnitudes and size has no impact on ViLT at all. Figure 3.42 shows the Log-Gaussian model fitted with ViLT's responses. Figure 3.43 shows the scalar variability model fitted with ViLT's responses.

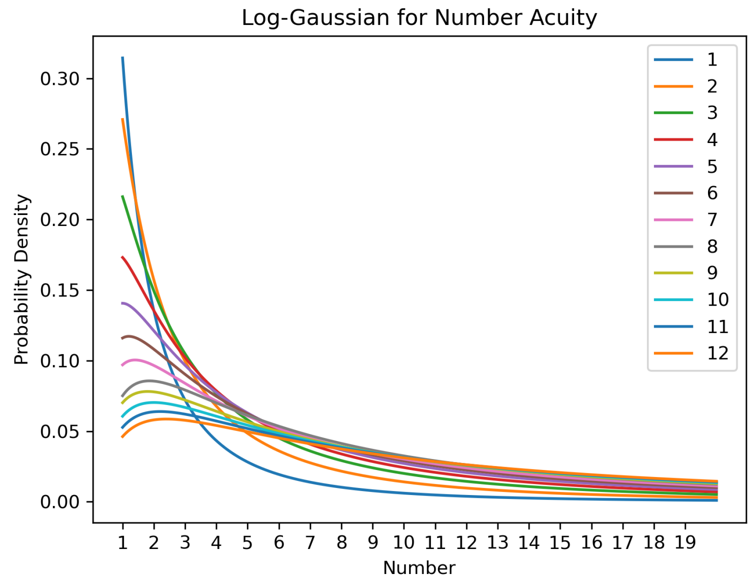

Figure 3.42: Log-Gaussian models with respect to 12 numbers for ViLT. Condition: congruent

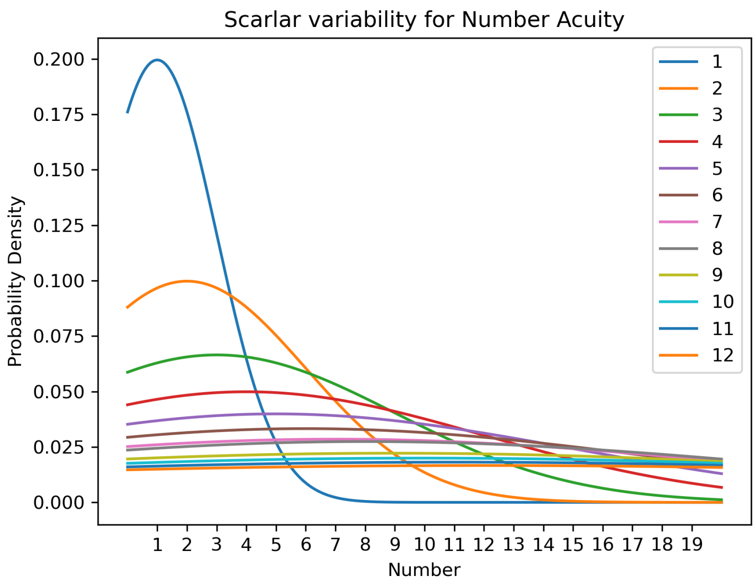

Figure 3.43: Scalar variability models with respect to 12 numbers for ViLT. Condition: incongruent

BLIP As for BLIP, it completely fails by answering all trails with ”left”. Thus, making it meaningless to calculate the Weber fraction or to fit the accuracy into mathematical models.

Discussion

4.1 Discrepancy between GPT-3.5 and GPT-4

Numerous studies have examined the mathematical capabilities of GPT-3.5 and GPT-4. In research conducted by OpenAI, GPT-3.5 demonstrated above-average mathematical abilities, while GPT-4 exhibited exceptional skills, surpassing 90% of human performance in some tests. It should be noted that these studies often use standardized benchmarks such as SAT Math and GRE Quantitative. For humans, a subject capable of complex higher mathematics, skillful use of calculus, and application of Fourier transformations to real problems would undeniably also easily solve basic math problems and likely possess a high level of number sense. However, the evaluation of GPT models' foundational mathematical abilities is insufficient.

These experiments reveal that although GPT-3.5 performs excellently on advanced mathematics tests, it does not perform as well on basic mathematics problems. Moreover, a study more specifically targeted at ChatGPT's mathematical capabilities demonstrated considerable inconsistency in ChatGPT's performance on complex mathematical problems.

GPT-4 significantly outperforms GPT-3.5 in many aspects, including mathematical abilities. Standardized tests indicate that GPT-4 has a score improvement of more than 10% over GPT-3.5, and adding the results of the experiments in this paper to the comparison, we can see that GPT-4 also excels in basic mathematical abilities.

Details about GPT-4's model structure, training data, and training methods are not available. However, through the behavioral patterns and results of this experiment, we can see that compared to GPT-3.5, GPT-4 has improved in two aspects when facing basic mathematical problems:

Improved calculation accuracy: In the tasks involving tokens and Arabic numerals, GPT-3.5 provided the correct calculation formula but made mistakes in the computation results.

Enhanced problem-solving strategy: GPT-4 demonstrates an improved approach to problem-solving, showing a clearer step-by-step methodology. This is particularly evident in complex problem sets where the solution requires multiple steps or the application of various mathematical principles. This process can be seen as an externalization of internalized mathematical representations into language. This externalization helps GPT-4 to better organize the steps and logical relationships in problem-solving.

Table 4.1: An example from the transcoding Tokens to Arabic Numerals, showing how GPT-3.5 performs more internally and less accurately

Example T2A

Prompt: We have three types of tokens. The big one corresponds to 200, the intermediate one corresponds to 5 and the small one corresponds to 1.
how to represent 47281 with these token(s)?

GPT-3.5: 47281 = (200 x 236) + (5 x 1) + (1 x 1)

GPT-4: To represent the number 47281 with these tokens, you need to determine how many of each token you would need. The best strategy is to use as many of the largest-valued tokens as possible, and then use the smaller-valued tokens to make up the remaining value. Here's the calculation:

1. First, calculate how many ”big” tokens (value 200) you can use.
‘47281 ÷ 200 = 236‘ (integer part, as you can't have a fraction of a token) with a remainder of ‘81‘.
So, you would need 236 ”big” tokens.
2. Next, calculate how many ”intermediate” tokens (value 5) you can use from the remainder of the previous step.
‘81 ÷ 5 = 16‘ with a remainder of ‘1‘.
So, you would need 16 ”intermediate” tokens.
3. Finally, calculate how many ”small” tokens (value 1) you can use from the remainder of the previous step.
‘1 ÷ 1 = 1‘.
So, you would need 1 ”small” token.

Therefore, to represent the number 47281, you would need 236 ”big” tokens, 16 ”intermediate” tokens, and 1 ”small” token.

Example A2T
Prompt: We have three types of colored tokens. The green one corresponds to 200, the orange one corresponds to 5 and the black one corresponds to 1. which number does it represent if we have 23 green token(s), 4 orange token(s), 2 black token(s).
GPT-3.5: Answer: 23 x 200 + 4 x 5 + 2 x 1 = 4,502
GPT-4: Each type of colored token represents a specific numerical value: Green token = 200 Orange token = 5 Black token = 1 If we have: 23 green tokens, 4 orange tokens, and 2 black tokens, we can calculate the total value represented by the tokens as follows: The total value of green tokens = 23 green tokens * 200 = 4600 The total value of orange tokens = 4 orange tokens * 5 = 20 The total value of black tokens = 2 black tokens * 1 = 2 By adding up these values, we can find the total number represented by all the tokens: Total number represented = Total value of green tokens + Total value of orange tokens + Total value of black tokens = 4600 + 20 + 2 = 4622 Therefore, 23 green tokens, 4 orange tokens, and 2 black tokens together represent the number 4622.

Table 4.2: An example from the transcoding Arabic Numerals to Tokens, showing how GPT-4 performs more internally and less accurately

Another reason for the improvement of GPT-4 could be simply the improvement of handling relatively long sequential data. By simply asking GPT-3.5 to calculate the pure calculation part (i.e. 23 x 200 + 4 x 5 + 2 x 1 = 4,502 in example A2T), GPT-3.5 can break down the calculation step by step and get the correct answer. It is reasonable to assume that the original lengthy prompt somehow interfered with the processing of the final calculation.

Table 4.3: An example of how GPT-3.5 fails

Pure Calculation
Prompt: 23 x 200 + 4 x 5 + 2 x 1 =
GPT-3.5: To calculate the expression 23 x 200 + 4 x 5 + 2 x 1, you can follow the order of operations (also known as PEMDAS or BODMAS) that states that multiplication should be done before addition. Here is the calculation: 23 x 200 + 4 x 5 + 2 x 1 = 4600 + 20 + 2 = 4620 + 2 = 4622 Therefore, 23 x 200 + 4 x 5 + 2 x 1 equals 4622.

4.2 Attention mechanism is not perfect

Both GPT-3.5 and GPT-4 fail in large number multiplication tasks, which is not a surprise, taking into account their intrinsic attention-based structure. Multiplying two 5-digit long numbers would get a number that is at least 11-digit long. These are long sequential data to handle. The bigger the number, the longer the sequence. GPT models can handle long sequences of letters that are much longer than 10 digits well if the tokenization is good but for long numbers, it is hard to tell that certain part of the sequence is more important to attend to. Needless to say, these sequences are very rare during the training phase, making this task even harder for neural networks. As a result, when the number becomes bigger and bigger, the model struggles more and more.

Transformer models, including those such as GPT-3.5 and GPT-4, employ an attention-based structure that is essential to their design and functionality. This attention-based structure enables these models to handle sequence data effectively by maintaining an internal state that captures the information from previous steps in the sequence and utilizes it for subsequent steps(Vaswani et al., 2017). However, while the transformer's attention-based nature can facilitate the learning of temporal dependencies, it can also pose challenges in handling long sequential calculations. One reason is that as the length of the sequence increases, the model's ability to maintain and utilize context information from the start of the sequence can diminish, a phenomenon often referred to as the vanishing gradient problem(Bengio et al., 1994). This can lead to errors in calculations that involve long sequences of numbers.

Another issue stems from the limited capacity of the transformer self-attention mechanism, which tends to give more weight to recent inputs in a sequence, potentially ignoring or downplaying the importance of earlier inputs(Rae & Razavi, 2020). In the context of long sequential calculations, this could mean that vital information from the start of the sequence is neglected, affecting the accuracy of the final calculation.

To mitigate these issues, more recent work has explored methods such as dynamic convolutions(Wu et al., 2019) and Transformer-XL(Dai et al., 2019), which incorporate mechanisms to better handle long-term dependencies.

4.3 Why Diffusion Models Fail

The four diffusion models tested in this thesis are an extension of the original diffusion model. They are text-conditioned diffusion models. To make the model conditioned on text, we typically modify the architecture of the model so that the denoising function can also take the conditioning text as input. The text is usually transformed into a fixed-size vector, for instance, by encoding it with a pre-trained language model (like GPT-2 or BERT), and this vector is then concatenated with the current state of the diffusion process at each step of the reverse process. In this way, the denoising function can use the information in the text to guide the noise removal process(Dhariwal & Nichol, 2021). The training of such a conditioned diffusion model calls for a data set consisting of image-text pairings. The model is subsequently trained to optimize the likelihood of the images given the corresponding texts, following the diffusion-generative process. This conditioning approach is not limited to images and can be extended to other data types such as audio and different conditioning information beyond text(Schölkopf et al., 2021).

In a conditioned diffusion model, image-text pairing serves as an important source of information that guides the model during its generation process. The conditioning text essentially provides a ”description” or ”specification” for the kind of image the model should produce. This is beneficial when the information conveyed by the text aligns with the level of detail depicted in the images. However, there are disparities in current training data between the volume of information manifested in text and images, especially when numerical content is involved. More specifically, numbers can be found in any given image, and they can exist in various forms, either as cardinal numbers used for counting or ordinal numbers indicating order. For example, the number 3 could represent the presence of 3 objects within an image or it could indicate that a certain object is the third from left to right. Numbers can exist in an image in various ways. It could be an Arabic '3'; a word 'three' or 3 objects that are in the same category. This is where numerosity perception diverges from other types of perception. While it depends on other sensory channels, it exhibits independence from other sensory channels during cognition. When applied, numerosity perception needs a certain degree of embodiment to be fully developed. Thus, the multimodal numerosity perception is no doubt harder for neural networks to possess.

Another possible reason why these four models fail may be the use of the attention mechanism. As a study shows, incorporating the encoder of large language models can significantly improve the performance of diffusion models(Saharia et al., 2022). DALL·E uses GPT-3 as an encoder(Ramesh et al., 2022) and Stable Diffusion uses a customized attention mechanism during training (Rombach et al., 2022). As a result, the text encoder may not specifically pay attention to the number, but rather to the main objects in the text.

4.4 Why ViLT & BLIP Fails

To understand why the models fail, we must look at what happened during the training phase of the models. As for training such models, the specifics would depend on the exact architecture and task, but typically it would involve:

Pre-training: In this phase, a large data set with visual and textual data will be used. For example, image-caption pairs from a data set like COCO could be used. The model would learn to associate the visual features of an image with the linguistic features of a caption. This is usually done in a self-supervised or semi-supervised way, for instance, by predicting masked parts of the image or caption.

Fine-tuning: In this phase, the pre-trained model would be fine-tuned on a specific task. For instance, if the task was visual question answering, the model could be fine-tuned on a data set like VQA v2 (e.g. ViLT). This usually involves supervised learning, where the model learns to predict the correct answer given the image and question.

The reason is similar to why diffusion models fail during the pre-training phase, the image-caption pairs together may not contain the same level of numerical information. The fine-tuning can also be due to the lack of numerical information. VAQv2 is a good data set with a wide range of questions. However, the questions in the data set can be too simple if the goal is to build a general and powerful multimodal network. Those questions can be answered with only a phrase or even a single word, such as 'yes' or 'no'. Numerical questions only require counting the number under 5 or repeating the Arabic Numerals shown on the image.

Neural networks can indeed show emergent abilities that are not shown during training, but this would only be possible if the model and the training material are large enough. ViLT which is fine-tuned only on VAQv2 and NLVR2 does not have the emergent ability.

4.5 Hints from ViLT and BLIP

Despite huge differences in the structure of the models. ViLT and BLIP achieve similar results. Unlike many other visual question-answering models, which use convolutional layers to extract visual features and use large language models (e.g. BERT, GPT) to extract textual features, ViLT encodes textual features with word embedding and encodes visual features with patch projection (i.e. linear embedding). These two ways of encoding are much simpler in terms of computational size, making ViLT a much more efficient model that can achieve equivalent levels of accuracy with a much shorter reasoning time (about 15ms). On the other hand, BLIP is built with a more complex structure. It encodes both image and text with a transformer-like encoder and then uses a transformer-like decoder to produce the output. Intuitively, I would consider that the structure of BLIP is more advanced and can contain more learned knowledge. However, that is not the case.

Take a closer look at the Geometrical Shape Recognition tasks with instructions in the prompt. Both models understand how to extract the key information to follow, that is, "answer with ___ triangle". Nevertheless, both models only understand that part and cannot generalize the example provided to other trials of tasks. That is to say, both models cannot align their behaviors well with human instruction. What makes ChatGPT so successful is that during its fine-tuning phase, it receives feedback from a human on how closely it behaves to the desirable behaviors desired by a human. This makes ChatGPT understand what the human wants and behave accordingly(Ouyang et al., 2022). It is possible that ViLT and BLIP cannot perform well on geometrical shape recognition tasks either after pre-training and fine-tuning on various larger data sets, even though training with larger data sets does entitle the models more abilities implicitly, but none of the models will be able to perform those abilities explicitly if they do not understand what the human wants. The same assumption could be drawn from the vision numerosity comparison task, where BLIP failed to align itself to instruction.

This could also be the reason why DALL·E is trained with GPT-3 (178M), which is much larger than the stable diffusion models' text encoder CLIP (33M) and only performs slightly better in counting and geometrical shape generation. Simply because, unlike ChatGPT, GPT-3 was not fine-tuned with reinforcement learning and thus could not understand instructions well.

4.6 Limitations and Future Works

In this thesis, some tests were run only once. Because these models can be deterministic or creative depending on the setting of parameters, the statistical results may not accurately describe the real competence of the models. Except for NADL-F, all other tests in this thesis are self-developed or lack statistical testing. That is to say, the reliability and variability of these self-developed tests are unknown.

Though this thesis tested some popular models' basic mathematical abilities in various ways, there are still many models out there to be tested. It is rational to develop a systematic way of implementing neural networks' basic mathematical abilities.

The image generation results are graded based on standards that are not wildly discussed and recognised in the academic world. Moreover, all results are only graded by one person and once.

More classic psychometric tests could be included in the test. With normative data from psychometric tests, we can easily compare the performance of neural networks and the performance of humans. By manipulating different parameters, it is possible to mimic the developmental process of neural networks' mathematical abilities.

Conclusion

In this dissertation, several neural network models were tested for their fundamental mathematical capabilities. Both GPT-3.5 and GPT-4 excelled at some basic tasks, with GPT-4 performing nearly flawlessly in all evaluations. Compared to GPT-3.5, GPT-4 demonstrated more precise computational abilities and a more nuanced logical reasoning process. Using simple prompting techniques, GPT-3.5 was also able to achieve results similar to GPT-4. In the standardized NADL-F tests, the performance of GPT-3.5 and GPT-4 was consistent with previous results.

In passive comprehension visual question-answering tasks, despite the drastically different model structures of ViLT and BLIP, their performance results were remarkably consistent. Both models did not excel in the recognition of geometric shapes in the basic plane and the counting. This could be due to the limited training data and the fact that the fine-tuning process during training did not endow the models with the ability to act according to human instructions.

In active expression counting abilities and plane geometric image generation tasks, the four different diffusion models were far from achieving the ideal accuracy rate. Even though DALL·E has shown good capability when numerosity ranges from 1 to 3. On the other hand, GPT models are good at both generating and counting textual materials given a certain numerosity.

Based on these findings, it can be concluded that, linguistically, GPT-3.5 and GPT-4 demonstrate good numerosity perception and number sense, and they are able to develop more complex mathematical abilities based on these skills. However, the other six models still need to make strides in improving their numerosity perception and number sense so that more complex mathematical abilities can come to light.

Bibliography

Agrillo, C., & Bisazza, A. (2018). Understanding the origin of number sense: A review of fish studies. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *373*(1740), 20160511.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). Vqa: Visual question answering. *Proceedings of the IEEE international conference on computer vision*, 2425–2433.

Arcara, G., Burgio, F., Benavides-Varela, S., Toffano, R., Gindri, P., Tonini, E., Meneghello, F., & Semenza, C. (2017). Numerical activities of daily living–financial (nadl-f): A tool for the assessment of financial capacities. *Neuropsychological Rehabilitation*.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Berch, D. B. (2005). Making sense of number sense: Implications for children with mathematical disabilities. *Journal of learning disabilities*, *38*(4), 333–339.

Brannon, E. M. (2005). What animals know about numbers. *Handbook of mathematical cognition*, *381*, 85–107.

Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

Burr, D., & Ross, J. (2008). A visual sense of number. *Current biology*, *18*(6), 425–428.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Dehaene, S. (2011). *The number sense: How the mind creates mathematics*. OUP USA.

Dehaene, S., Dupoux, E., & Mehler, J. (1990). Is numerical comparison digital? analogical and symbolic effects in two-digit number comparison. *Journal of experimental Psychology: Human Perception and performance*, *16*(3), 626.

Dehaene, S., Piazza, M., Pinel, P., & Cohen, L. (2003). Three parietal circuits for number processing. *Cognitive neuropsychology*, *20*(3-6), 487–506.

Dhariwal, P., & Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, *34*, 8780–8794.

Du, Y., Durkan, C., Strudel, R., Tenenbaum, J. B., Dieleman, S., Fergus, R., Sohl-Dickstein, J., Doucet, A., & Grathwohl, W. (2023). Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. *arXiv preprint arXiv:2302.11552*.

Feigenson, L., Dehaene, S., & Spelke, E. (2004). Core systems of number. *Trends in cognitive sciences*, *8*(7), 307–314.

Frieder, S., Pinchetti, L., Griffiths, R.-R., Salvatori, T., Lukasiewicz, T., Petersen, P. C., Chevalier, A., & Berner, J. (2023). Mathematical capabilities of chatgpt. *arXiv preprint arXiv:2301.13867*.

Ganesh, P., Volle, K., Burks, T., & Mehta, S. (2019). Deep orange: Mask r-cnn based orange detection and segmentation. *IFAC-PapersOnLine*, *52*(30), 70–75.

Gebuis, T., Kadosh, R. C., & Gevers, W. (2016). Sensory-integration system rather than approximate number system underlies numerosity processing: A critical review. *Acta psychologica*, *171*, 17–35.

Gersten, R., & Chard, D. (1999). Number sense: Rethinking arithmetic instruction for students with mathematical disabilities. *The Journal of special education*, *33*(1), 18–28.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Halberda, J., Mazzocco, M. M., & Feigenson, L. (2008). Individual differences in non-verbal number acuity correlate with maths achievement. *Nature*, *455*(7213), 665–668.

Jordan, N. C., Glutting, J., & Ramineni, C. (2010). The importance of number sense to mathematics achievement in first and third grades. *Learning and individual differences*, *20*(2), 82–88.

Jordan, N. C., Kaplan, D., Locuniak, M. N., & Ramineni, C. (2007). Predicting first-grade math achievement from developmental number sense trajectories. *Learning disabilities research & practice*, *22*(1), 36–46.

Kaufman, E. L., Lord, M. W., Reese, T. W., & Volkmann, J. (1949). The discrimination of visual number. *The American journal of psychology*, *62*(4), 498–525.

Kim, W., Son, B., & Kim, I. (2021). Vilt: Vision-and-language transformer without convolution or region supervision. *International Conference on Machine Learning*, 5583–5594.

Knuth, D. E. (1974). Computer science and its relation to mathematics. *The American Mathematical Monthly*, *81*(4), 323–343.

Koubaa, A. (2023). Gpt-4 vs. gpt-3.5: A concise showdown.

Kriegeskorte, N. (2015). Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annual review of vision science*, *1*, 417–446.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, *40*, e253.

Lample, G., & Charton, F. (2019). Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Li, J., Li, D., Xiong, C., & Hoi, S. (2022). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *International Conference on Machine Learning*, 12888–12900.

Linn, M. C., & Petersen, A. C. (1985). Emergence and characterization of sex differences in spatial ability: A meta-analysis. *Child development*, 1479–1498.

McClelland, J. L., Rumelhart, D. E., Group, P. R., et al. (1986). *Parallel distributed processing* (Vol. 2). MIT press Cambridge, MA.

Newcombe, N. S., Weisberg, S. M., Atit, K., Jacovina, M. E., Ormand, C. J., & Shipley, T. F. (2015). The lay of the land: Sensing and representing topography. *Baltic International Yearbook of Cognition, Logic and Communication*, *10*(1), 6.

Nieder, A. (2016). The neuronal code for number. *Nature Reviews Neuroscience*, *17*(6), 366–382.

OpenAI. (2022a). *Introducing chatgpt*. https://openai.com/blog/chatgpt

OpenAI. (2022b). *Introducing dall·e*. https://openai.com/dall-e-2

OpenAI. (2023). Gpt-4 technical report.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback.

Piazza, M., Facoetti, A., Trussardi, A. N., Berteletti, I., Conte, S., Lucangeli, D., Dehaene, S., & Zorzi, M. (2010). Developmental trajectory of number acuity reveals a severe impairment in developmental dyscalculia. *Cognition*, *116*(1), 33–41.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Rae, J. W., & Razavi, A. (2020). Do transformers need deep long-range memory. *arXiv preprint arXiv:2007.03356*.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Riley, K. F., Hobson, M. P., & Bence, S. J. (1999). Mathematical methods for physics and engineering.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695.

Rumelhart, D. E., Hinton, G. E., McClelland, J. L., et al. (1986). A general framework for parallel distributed processing. *Parallel distributed processing: Explorations in the microstructure of cognition*, *1*(45-76), 26.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, *35*, 36479–36494.

Sarkar, A., Chowdhury, T., Murphy, R., Gangopadhyay, A., & Rahnemoonfar, M. (2023). Sam-vqa: Supervised attention-based visual question answering model for post-disaster damage assessment on remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117.

Schmidt, J., Marques, M. R., Botti, S., & Marques, M. A. (2019). Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, *5*(1), 83.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., & Bengio, Y. (2021). Toward causal representation learning. *Proceedings of the IEEE*, *109*(5), 612–634.

Shakarian, P., Koyyalamudi, A., Ngu, N., & Mareedu, L. (2023). An independent evaluation of chatgpt on mathematical word problems (mwp). *arXiv preprint arXiv:2302.13814*.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *International Conference on Machine Learning*, 2256–2265.

Song, J., Meng, C., & Ermon, S. (2020). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Starkey, P., & Cooper Jr, R. G. (1980). Perception of numbers by human infants. *Science*, *210*(4473), 1033–1035.

Starr, A., Libertus, M. E., & Brannon, E. M. (2013). Number sense in infancy predicts mathematical abilities in childhood. *Proceedings of the National Academy of Sciences*, *110*(45), 18116–18120.

Stoianov, I., & Zorzi, M. (2012). Emergence of a'visual number sense'in hierarchical generative models. *Nature neuroscience*, *15*(2), 194–196.

Testolin, A., Zou, W. Y., & McClelland, J. L. (2020). Numerosity discrimination in deep neural networks: Initial competence, developmental refinement and experience statistics. *Developmental science*, *23*(5), e12940.

Trick, L. M., & Pylyshyn, Z. W. (1994). Why are small and large numbers enumerated differently? a limited-capacity preattentive stage in vision. *Psychological review*, *101*(1), 80.

Uttal, D. H., Miller, D. I., & Newcombe, N. S. (2013). Exploring and enhancing spatial thinking: Links to achievement in science, technology, engineering, and mathematics? *Current Directions in Psychological Science*, *22*(5), 367–373.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, M., & Deng, J. (2020). Learning to prove theorems by learning to generate theorems. *Advances in Neural Information Processing Systems*, *33*, 18146–18157.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, *35*, 24824–24837.

Wu, Z., Shen, C., & Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, *90*, 119–133.

Zheng, C., Liu, Z., Xie, E., Li, Z., & Li, Y. (2023). Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Zhou, T., Wang, S., Zhou, Y., Yao, Y., Li, J., & Shao, L. (2020). Motion-attentive transition for zero-shot video object segmentation. *Proceedings of the AAAI conference on artificial intelligence*, *34*(07), 13066–13073.

Appendix

6.1 Code Example

```
## Key functions ##
def get_result(openai_creation):
    conv2json=to_json_schema(openai_creation)
    convedjson=conv2json['choices']
    dict=convedjson[0]
    result=dict['text']
    return result

def get_index(key_word,prompt):
    index=prompt.find(key_word)
    return index

def prompt_generator(var_ls,origin_prompt,*index):
    prompt_local=[]
    if len(index)==1:
        index=index[0]
        for number in var_ls:
            number=str(number)
            prompt=origin_prompt[:index]+' '+number+' '+origin_prompt[index
                :]
            prompt_record.append(prompt)
            prompt_local.append(prompt)
        return prompt_local
```

```
    elif len(index)==2:
        idx1,idx2=index[0],index[1]
        var1,var2=var_ls[0],var_ls[1]
        for (num1,num2) in zip(var1,var2):
            num1,num2=str(num1), str(num2)
            prompt=origin_prompt[:idx1]+' '+num1+' '+origin_prompt[idx1:
                idx2]+' '+num2+' '+origin_prompt[idx2:]
            prompt_record.append(prompt)
            prompt_local.append(prompt)
        return prompt_local

    elif len(index)==3:
        idx1,idx2,idx3=index[0],index[1],index[2]
        var1,var2,var3=var_ls[0],var_ls[1],var_ls[2]
        for (num1,num2,num3) in zip(var1,var2,var3):
            num1,num2,num3=str(num1), str(num2),str(num3)
            prompt=origin_prompt[:idx1]+' '+num1+' '+origin_prompt[idx1:
                idx2]+' '+num2+' '+origin_prompt[idx2:idx3]+' '+num3+' '+
                origin_prompt[idx3:]
            prompt_record.append(prompt)
            prompt_local.append(prompt)
        return prompt_local

def looper(prompt_ls,max_token):
    aiobject=[]
    for prompt in prompt_ls:
        json_result=openai.Completion.create(model="text-davinci-003",
        prompt=prompt,
        max_tokens=max_token,
        temperature=0.1)
        aiobject.append(json_result)
    for result in aiobject:
        result=get_result(result)
        final_result.append(result)
## a short example ##
```

```
ara_token=[2,21,3,201,22,130,32,212,51,150,43,313,
8895,2758,10002,47281,88988]
ara_token_prompt='''We have three types of tokens. The green one
    corresponds to 200, the orange one corresponds to 5 and the black one
    corresponds to 1.
how to represent with these token(s)?'''
idx=get_index('with these token(s)',ara_token_prompt)
at_prompts=prompt_generator(ara_token,ara_token_prompt,idx)
looper(at_prompts,256)
```

6.2 Notebooks

All Jupyter notebooks can be found here.

6.2.1 Number Screening

https://colab.research.google.com/drive/1xC0cjSrH1XCoSk8phHKWKrLGmDS83tJQ?usp=sharing

6.2.2 Number Screening with customized data set

https://colab.research.google.com/drive/1YWknKZSLebnBte-Tce4PL5KbVYEk-1NO?usp=sharing

6.2.3 NADL-F

https://colab.research.google.com/drive/12z4TQ5wcqMSsXuaEyY43B-Xj8I9Nw3fH?usp=sharing

6.2.4 Diffuser models

https://colab.research.google.com/drive/1OB_DmStjZ40pbyAAhdoteOSA886Qz9Wz?usp=sharing

6.2.5 ViLT

https://colab.research.google.com/drive/1idydzsbZDtGqZjwy6p1BBVIiQXbtPsEN?usp=sharing

6.2.6 BLIP

https://colab.research.google.com/drive/1SmZQ4H8fB0TuzsJw9cOKpC5kqy1koPfy?usp=sharing

6.2.7 Geometrical pics

For generating all the plane geometrical shapes and other images used in this thesis https://colab.research.google.com/drive/1tLRO8LdWLQUNMSkKV8gp733ZVyvzLMHU?usp=sharing

6.2.8 GPT models in Counting and Generation tasks

https://colab.research.google.com/drive/1ff5Z7OIoPfRnzZIeEumFgcdK83iUiaHn?usp=sharing

6.3 Materials

Images generated by diffusion models. https://drive.google.com/drive/folders/1phtF63tE1lmDiTtMseAHJBtbZh64fBd-?usp=sharing

6.4 Standard for Grading Image Generation Results

Click here

6.5 Detailed Responses

Detailed responses of each model in all tests of this thesis can be requested by leaving a comment here.