A Vision Check-up for Language Models

Pratyusha Sharma[*1] Tamar Rott Shaham[*1] Manel Baradad[1] Stephanie Fu[†2]
Adrián Rodríguez-Muñoz[1] Shivam Duggal[1] Phillip Isola[1] Antonio Torralba[1]
[1]MIT CSAIL [2]UC Berkeley

Abstract

What does learning to model relationships between strings teach Large Language Models (LLMs) about the visual world? We systematically evaluate LLMs' abilities to generate and recognize an assortment of visual concepts of increasing complexity and then demonstrate how a preliminary visual representation learning system can be trained using models of text. As language models lack the ability to consume or output visual information as pixels, we use code to represent images in our study. Although LLM-generated images do not look like natural images, results on image generation and the ability of models to correct these generated images indicate that precise modeling of strings can teach language models about numerous aspects of the visual world. Furthermore, experiments on self-supervised visual representation learning, utilizing images generated with text models, highlight the potential to train vision models capable of making semantic assessments of natural images using just LLMs.

1. Introduction

What does it mean to understand the visual concept of *e.g.* a "frog"? Is it sufficient to know the color of its skin, the number of feet it has, the location of its eyes, details about how it looks when it hops? While a subset of this information about the appearance of a frog can be acquired from text, it is widely believed that to understand the concept visually, one would need to observe an image of a frog or, better still, observe it from different perspectives and in various real-world scenes. However, to what extent can we learn about the visual "meaning" of different concepts from text[1]?

Despite operating on textual data, language model representations have been shown to contain information about named perceptual concepts like shape and color [9, 22, 36] and have been found to be linearly translatable to representations learned by vision models [27, 37]. These experiments demonstrate that independently, vision and language models represent aspects of the world similarly. While investigating model representations for a pre-selected set of attributes can inform us about information encoded by the model, it limits studying a fixed set of attributes at once and requires access to the model's internal parameters. Instead, as seen in Fig. 1, we ask:

1. *What do language models know about the visual world?*
2. *Can we train a vision system for natural images using a text-only model?*

To answer these questions, we evaluate what information about the visual world off-the-shelf language models contain by testing their ability to render *(draw)* and recognize *(see)* real-world visual concepts. This allows us to measure their ability to model arbitrary properties, both individually and concurrently, without training a classifier for a rigid set of features one at a time. Although LLMs are limited in their ability to generate images using pixels, examples presented by [8] suggest that models like GPT-4 can generate code capable of rendering objects like a unicorn. We take this further by measuring LLMs abilities to generate visual concepts of increasing complexity via a *textual prompt* $\rightarrow$ *code* $\rightarrow$ *image* procedure. Figure 2 shows examples of complex scenes generated by LLMs. We find that LLMs are surprisingly good at generating intricate visual scenes composed of multiple objects, effectively modeling spatial relations between them. However, there are aspects of the visual world that LLMs fail to capture, including objects' properties like their textures, precise shapes, as well as surface contact with other objects in the image.

Next, we evaluate the ability of LLMs to recognize (see) perceptual concepts (Fig. 1 Part I (b)). We do this by collecting human drawings represented as code and asking LLMs *what* they depict. The code describes the ordered sequence of shapes, their locations, and colors. We find that unlike humans, where creation is hard and verification is easy, models struggle to interpret/recognize code describing detailed scenes that they themselves can effectively generate.

Further, we demonstrate that the visual generation competence of a language model can be improved using text-

[*]Indicates equal contribution.
[†]Work done while at MIT CSAIL.
Project page: https://vision-checkup.github.io/
[1]While the question of what can be learned about the visual world from natural language alone is interesting, in our case, "text" represents the space of all strings on the internet, including natural language and code.

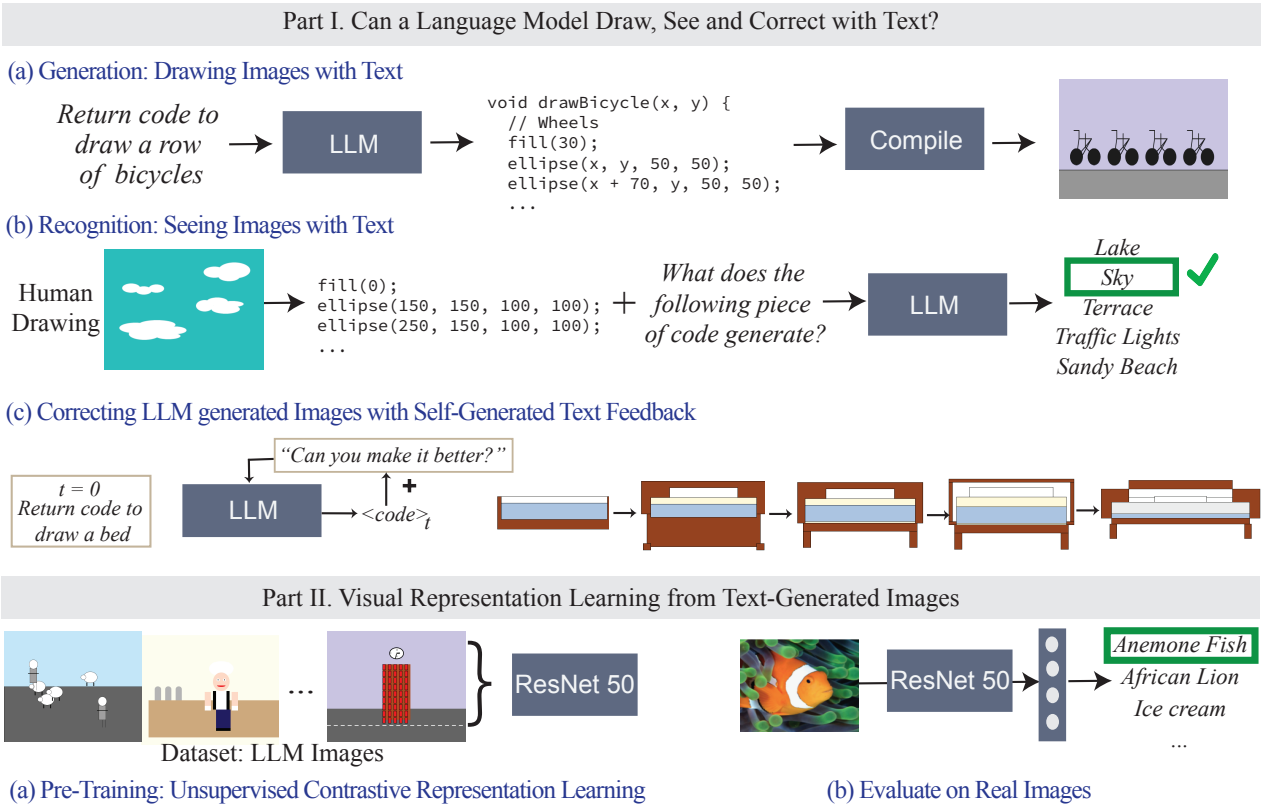


Figure 1. **Vision check-up for LLMs.** I. Testing the visual knowledge of Language Models. We suggest a set of tests to check the vision abilities of language models, these include (a) the ability to write code that renders complex visual concepts (b) recognizing visual concepts from code (c) correcting rendering code with text-only self-feedback. II. We test whether LLMs can generate data to train a high-performance vision system that can be used to make semantic judgments on natural images.

based corrections. We do this by closing the feedback loop between the LLMs and itself. Here, we first use the language model to generate code illustrating a concept. Following that, the model is repeatedly called by conditioning its generation on its previously generated code and prompted to "improve its generated code". We find that making such iterative calls to the model results in improved visual depictions, as shown in Fig. 1 (Part I (c)).

Finally, we study if LLM-generated images could serve as a data source for pre-training vision models and compare them to synthetically generated and natural images. By constructing a pre-trained visual representation system from *only* text that transfers well to tests on natural images, we demonstrate that text models capture aspects of the visual world similar to those present in natural images.

To summarize, the paper's main contributions are:

1. **The Visual Aptitude Dataset**: Introducing a hierarchical visual categories dataset consisting of shapes, objects, and scenes descriptions to test the visual capabilities of language models.
2. **Generation**: Testing and quantifying the generation capabilities of LLM's, showing that it is possible to generate detailed and diverse scenes using text-only models. We also show that it is possible to improve the quality of the generated images by using text-based feedback.
3. **Recognition:** Analyzing whether LLMs are also able to recognize image generative code as well as producing it. We test this capability using out-of-distribution samples generated by humans, which we crowdsource.
4. **Training for natural image tasks without natural images**: We show that the images generated by LLMs are useful for training visual backbones, achieving state-of-the-art performance when complemented with other procedurally generated image datasets.

2. Related work

Vision and language models: Language models have been shown to be extremely powerful in understanding and generating visual information when paired with vision models [42], training vision adaptors [15, 24, 27, 49], or when trained jointly over visual and textual data [37, 40, 43]. Although vision-language pre-training / chaining vision and language models allow models to reason about aspects of the visual world, we investigate the visual capabilities of models representing images with text. Furthermore, several benchmarks have been proposed to evaluate the ability of LLMs on textual tasks [16, 18, 20, 25, 34]. Unlike them, we propose a procedure to evaluate LLM's *vision* abilities.

Visual Understanding in Language Models. Meaning

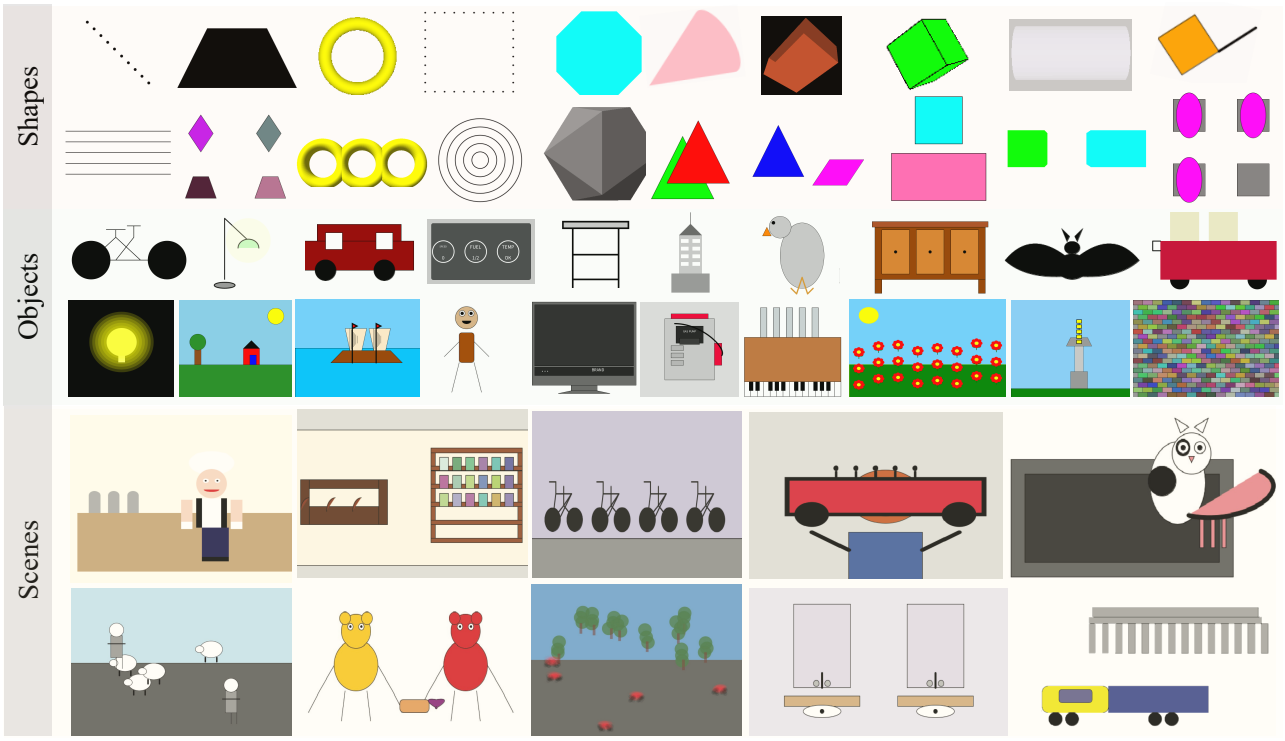


Figure 2. **Visual Aptitude Dataset.** We collect a dataset of visual concepts of including shapes, objects and scenes, and ask LLMs to generate corresponding images using a Text → Code → Image generation procedure. Guess the captions of the scenes![2]

Representations [9, 22, 36] show that language models contain information of the state of the world in their internal representations that can be recovered by probing these models for attributes of interest. Additionally, [2, 28, 41] demonstrate that language models are capable of representing visual concepts such as "color" and "shape". However, the attributes studied have been limited to only those that can be described by natural language and can only be investigated one at a time. Moreover, with models being placed behind closed walls, it becomes increasingly difficult to probe the internal representation of models for the presence of visual attributes of interest.

Program synthesis via LLMs. Pioneered by OpenAI's Codex [10], Github's Copilot [1], Meta's Code Llama [32] etc., LLMs have been shown to possess exceptional coding capabilities [46]. Recently, Bubeck *et al*. [8] highlighted the emergent properties of a GPT-4 model for image generation / scalable vector graphics via text-prompted TikZ or javascript codes. In this work, we build on their insights and carefully examine the diversity and realism of multiple text-only language models like GPT-3.5, Davicini, GPT-4 (see Fig. 3, Fig. 5) for programmable image generations. Furthermore, as one of our key contributions, we analyze the usefulness of the procedurally generated images for self-supervised visual representation learning (see Sec. 5).

Training vision models with synthetic data. The ability to train a vision system using synthetic data was studied in several tasks including optical flow [7, 14, 26], segmentation [12, 30], detection [31], classification [4, 33], and representation learning [39]. Perhaps the most related set of work studied how to train vision backbones using images generated from human-written code which capture different priors of the visual world, such as textures and shapes. These include generative processes like fractals [3, 21], dead leaves [6], sinusoidal waves [35], and even a crowd-sourced dataset of thousands of generative programs [5]. While these approaches achieve competitive performance, it remains unclear how to systematically introduce high-level semantic concepts related to shapes and scene composition without the intervention of human experts.

3. Visual Aptitude Dataset: Points to Scenes

We evaluate an LLM's visual competence by measuring its ability to create, recognize, and modify image-rendering code on a hierarchy of concepts. This resulting dataset of images also serves as the corpus used for pre-training a vision model in the later part of the paper. We construct three

[2] **Captions for Fig 2 scenes:** (left to right, top to bottom) **C**hef standing next to a counter with jars; **O**ffice with leather couch, surrounded by books; **R**ow of bicycles; **B**irthday boy with car shape cake & candles; **B**lack & white cat sitting on side of a computer monitor; **C**ouple of men hearding sheep down the road; **T**wo stuffed animals cutting bread & spreading jelly on it; **B**lurred image of motorized scooters on wooded road. **B**athroom with two sinks & tow mirrors. **Y**ellow & blue train is next to an overhang;

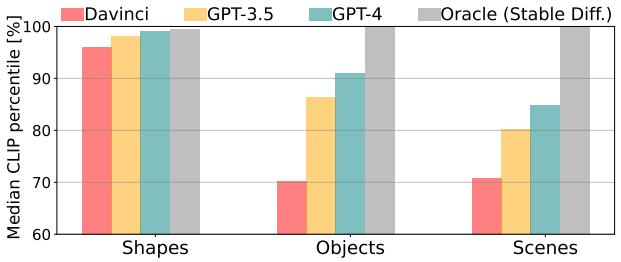


Figure 3. **Image-Text Fidelity.** Median CLIP image-text retrieval percentiles of images generated by different LLMs. We include Stable Diffusion as an Oracle. Chance is 50%.

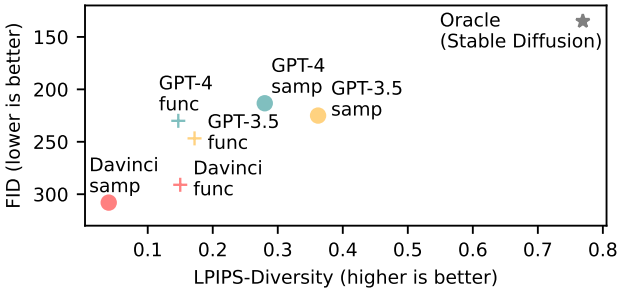


Figure 4. **Realism vs. Diversity.** With both sampling strategies, LLMs are able to draw diverse illustrations of the same concept.

datasets with textual descriptions of visual concepts of gradually growing complexity. Starting with simple shapes and their compositions, to objects, and finally to complex scenes described by elaborate natural language descriptions. Samples from the dataset can be found in Fig. 2 and in the Supplementary Material (SM).

(i) *Shapes and their compositions:* The first dataset contains a composition of shapes from different categories such as points, lines, 2D-shapes, and 3D-shapes with 32 different attributes like color, texture, location, and spatial arrangement. The full dataset contains more than 400K examples, and we sample 1500 for tests in our experiments.

(ii) *Objects:* The second dataset contains the 1K most frequent objects of the ADE20K dataset [47, 48]. Objects are more difficult to generate and recognize than shapes, as they contain complex compositions of many shapes.

(iii) *Scenes:* The last dataset consists of complex scene captions describing diverse places with multiple objects. For this, we uniformly sample 1000 scene descriptions at random from the MS-COCO [23] dataset.

3.1. Representing images with code

In the dataset, the visual concepts are described with language. For instance, we can describe a scene as "a sunny summer day on a beach, with a blue sky and calm ocean." We test the visual competence of LLMs by prompting them with these descriptions and measuring their ability to generate code that can be compiled to render images depicting the scenes. Why code? While LLMs can sequentially output pixel values to generate images [8] their ability to do so is currently limited. Code, on the other hand, can provide a descriptive yet concise representation of the visual world. It can be used to represent higher-level perceptual attributes and language models are already trained on examples of code. In this paper, code in different programming languages will serve as the primary mode of interaction and visual scene generation for LLMs.

3.2. Models and Programming Languages tested

For this study, we evaluate four language models, each tested on four different programming languages.

(i) *Language models:* We evaluate the visual competence of GPT-3.5 (both `text-davinci-003` and `GPT-3.5-turbo` models) and GPT-4[3]. Models, like Llama2 (chat 70B), GPT-J, and GPT-2 failed to generate executable image-rendering code reliably and are excluded from analysis in the main paper but are included in the SM.

(ii) *Programming languages.* To validate that a model's visual capability is not restricted to a specific programming language we use four programming languages with different expressiveness. These are: python-matplotlib, python-turtle, Processing (built over Java), and TikZ (built over Tex). A model's ability to generate and recognize the same visual concept across programming languages hints at the model possibly having a coherent and language-agnostic representation of that concept.

4. A Vision Checkup for LLMs

In this section, we evaluate the visual capabilities of LLMs. The models are evaluated on three tasks: (i) Generation / Drawing with text: assesses an LLM's competence in generating image-rendering code corresponding to a specific concept. (ii) Recognition / Seeing with text: tests the LLMs's performance in recognizing visual concepts and scenes represented as code. We test each model on the code representation of human drawings. (iii) Correcting drawings with text feedback: evaluates an LLM's ability to iteratively modify its generated code using natural language feedback generated by itself.

4.1. Generation: Drawing with Text

Experimental Setup. To test what LLMs are capable of visualizing, we evaluate their ability to generate code representing concepts from our Visual Hierarchy dataset across four programming tools. The prompt fed to the LLM is:

Prompt: *"write code in the programming language "[programming language name]" that draws a [concept]"*.

We then render the images by compiling the code in its corresponding programming language. Additional details about the experiment protocol can be found in the SM.

[3]There are two types of GPT-4. We interact here with GPT-4 model and not the GPT-4(V) model.

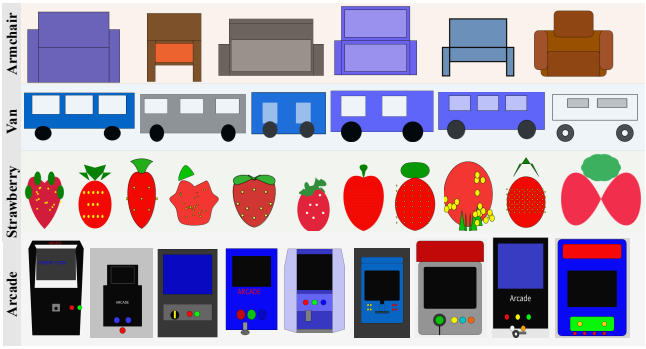


Figure 5. **Diversity.** LLMs are capable of generating diverse meaningful instances of the same concept, showcasing their ability to represent concepts beyond a single fixed prototype.

Evaluation protocol. We evaluate the visual quality and diversity of the images rendered using the following metrics.

(1) *Fidelity*: We compute the fidelity between the generated image and its ground-truth caption by retrieving the best caption for the image. We first calculate the agreement between each image and all potential captions within the same category (shapes/objects/scenes) using their CLIP score. We then report the rank of the ground-truth caption in percentage (*e.g.* a score of 100% implies that the ground-truth concept is ranked first). In the SM, we augment this measure with human perceptual studies and show that the CLIP rank reflects human preference as evidenced from their high correlation ($r = 0.82$,p-val$= 1.25e^{-09}$).

(2) *Diversity:* To assess the ability of the models to render diverse content, we use the LPIPS-diversity score [45] over image pairs representing the same visual concept.

(3) *Realism:* For a uniformly sampled collection of 1K images from ImageNet [13], we use the Fréchet Inception Distance (FID) [19] to quantify the difference in the distribution of natural images and those generated by LLMs.

Baseline: As an oracle, we include images generated by a text-to-image model (Stable Diffusion [29]) and report their scores across all evaluation metrics.

What can LLMs visualize? We find that LLMs can visualize real-world concepts from across the visual hierarchy. Examples of LLM-rendered images can be found in Fig. 1, 2, 5, and in the SM. LLMs are capable of generating non-trivial visual compositions, examples of such are shown in Fig. 2; The model composes two unrelated concepts ("car shaped cake"), generates visual phenomena ("blurred image"), and manages to correctly interpret spatial relations (*e.g.* "a row of bicycles" arranged horizontally.) Unsurprisingly, the competence of the models deteriorates with increasing concept complexity from shapes to scenes, as seen in the median image-text retrieval CLIP-scores across the different categories Fig. 3. For more complex visual concepts such as drawing scenes comprising multiple objects, GPT-3.5 and GPT-4 are more accurate at drawing scenes with intricate descriptions using processing and tikz than python-matplotlib and python-turtle. For objects and scenes, CLIP scores indicate that concepts containing "person", "vehicle", and "outdoor scenes" are the easiest to draw (see full analysis in the SM). This ability to render complex scenes comes from the expressivity of the rendering code, the model's programming capability in each of them, and the quality of its internal representations of the different concepts involved.

Model Name	Objects	Scenes
Davinci	0.136	0.221
GPT3.5	0.228	**0.380**
GPT4	**0.234**	0.212
Baseline [Chance]	0.2	0.2

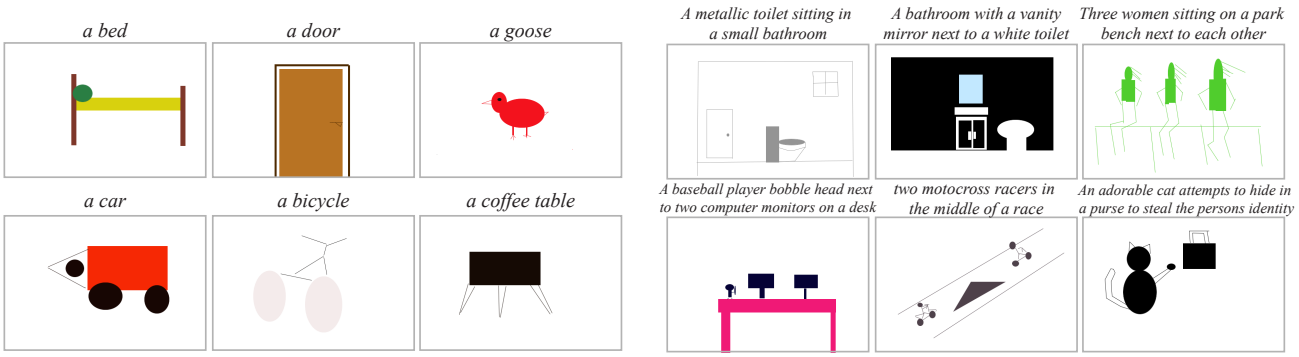
Table 1. **Recognition of Human Drawings.** Models struggle to correctly classify human drawings into their categories. While GPT-3.5 correctly classifies images over chance for the scenes category, other models classify images correctly barely over chance.

What can LLMs not visualize? In some cases, even relatively simple concepts are difficult for the models to draw. We identify several common failure modes: (a) Language models specifically struggle with concepts combining a set of shapes and a specific spatial organization, (b) Drawings are coarse and lack detail. These are the common failure cases for Davinci, especially when coding with matplotlib and turtle. (c) Depiction is incomplete, corrupted, or represents only a subset of the concepts (typical for the scenes category). An interesting standalone failure case is drawing digits. With all models and languages, we found this task to be challenging. See SM for a discussion of failure cases and the effect of prompting on the model's generation.

Diversity and Realism. Language models exhibit the ability to generate diverse visualizations of the same concept as seen in Fig. 5. To generate different samples of the same scenes, we compare two strategies: (i) Repeated sampling from the model (ii) Sampling a parametrized *function* that allows creating a new drawing of the concept by changing parameters. The ability of the model to render diverse realization of visual concepts is reflected in the high LPIPS-diversity scores in Fig. 4. The ability to generate diverse images suggests that LLMs can represent the visual concept in many ways rather than a limited set of prototypes. LLM generated images are far from as realistic as natural images, with the models scoring poorly on the FID metric as compared to the Stable Diffusion oracle (Fig 4). However, it is interesting to note that modern models rank better than older models, indicating that they might be slowly inching towards increasingly realistic representations of the world.

4.2. Recognition: Seeing with Text

Recognizing the contents of an image requires inferring how elements such as points, strokes, colors, and shapes



(a) Objects (b) Scenes

Figure 6. **Human drawings.** Examples of drawings made by users using our drawing interface that passed the CLIP score filtering. Each of the collected drawings is converted into processing code and is then included in the LLM's recognition tests.

combine spatially to give rise to objects that are themselves composed to represent complex scenes. Evaluating a model's ability to recognize image-rendering code offers insights into its competence in interpreting high-dimensional representations of visual scenes, including ones very different from its "memorized" prototypes. While code on the internet can be a good source to test these models, it contains comments, print statements, and symbol names that reveal essential semantic information about what the code represents and may not actually require sophisticated inference over the steps used to create the drawing. Further, we wanted to ensure that the code used for evaluation recognition was new and not part of the training set for any of the models tested. Therefore, we collect our own dataset of code representing drawings as seen in Fig. 6.

Dataset. While most people excel at drawing images using "Paint"-like applications, writing code that represents images is trickier and requires more skill. To test the recognition ability of models we collect a dataset of human drawings and their corresponding representation in code. This dataset was constructed following a setting similar to the game Pictionary or Google's "Quick Draw!" application. Users were prompted to draw a concept from the Visual Aptitude Dataset within a two-minute time limit using a drawing interface. The interface recorded their drawing steps in the Processing programming language. Shapes were excluded from the analysis, and the generated images were filtered for quality using the Fidelity score. The interface was designed to mimic the components expressed in drawings made with processing and to encourage users to portray the entire prompt rather than a detailed but incomplete illustration in the allotted time. We collect 600 images per category, of which 162 object images and 113 scene images pass the fidelity check (rank$\leq$ 40) and are used for evaluation. For further details, refer to the SM.

Evaluation protocols. We evaluate the models' capability of recognizing visual concepts represented as code by measuring the model's ability to classify the image in a *multi-class classification* setting. In this test, the model is prompted with the code and a list of visual concepts, where one of the list's entries describes the code accurately. The model is then tasked with matching the visual concept to the correct entry. The prompt to the model is:

Prompt: *"Which of the following concepts from the list does the [code] represent? [concept1, concept2,...]".*

Baseline: The success of the chance baseline is decided by the probability of the desired outcome. This is given by $1/N$ where N is the number of outcomes. For the multi-class classification setting with five labels this is 0.2.

Analysis. Human drawings present a unique recognition challenge as there is a lot of diversity in the images representing a given concept.

Language models can do very limited spatial reasoning. Table. 1 shows that GPT-3.5 beats the chance baseline across the scenes setting, demonstrating that the visual recognition capability of the models is non-trivial, allowing it to (limitedly) recognize code representing human drawings. While the exact mechanism that allows models to do so is unclear, the task requires models to identify objects, their attributes and spatial arrangements. Wrongly attributing any information can result in a completely different visual scene. Although models cannot perfectly recognize all images, the exhibited recognition capability is non-trivial.

Models can fail to recognize concepts they can otherwise draw very well. Unlike humans, where the ability to draw something well automatically implies the ability to recognize the same concept well, models can fail to recognize concepts they have no problem rendering. This contradicts the notion that creation is hard, but verification can be easy. Failures in recognition, as shown in the SM, showcase that images that models fail to recognize are the non-stereotypical depictions of the visual concept, showcasing that there might be limits to a model's ability to recognize perceptual concepts from code.

4.3. Textual-Feedback: Correcting with Text

The model's ability to generate an image is limited in part by the prompt. Failure to generate an image corresponding

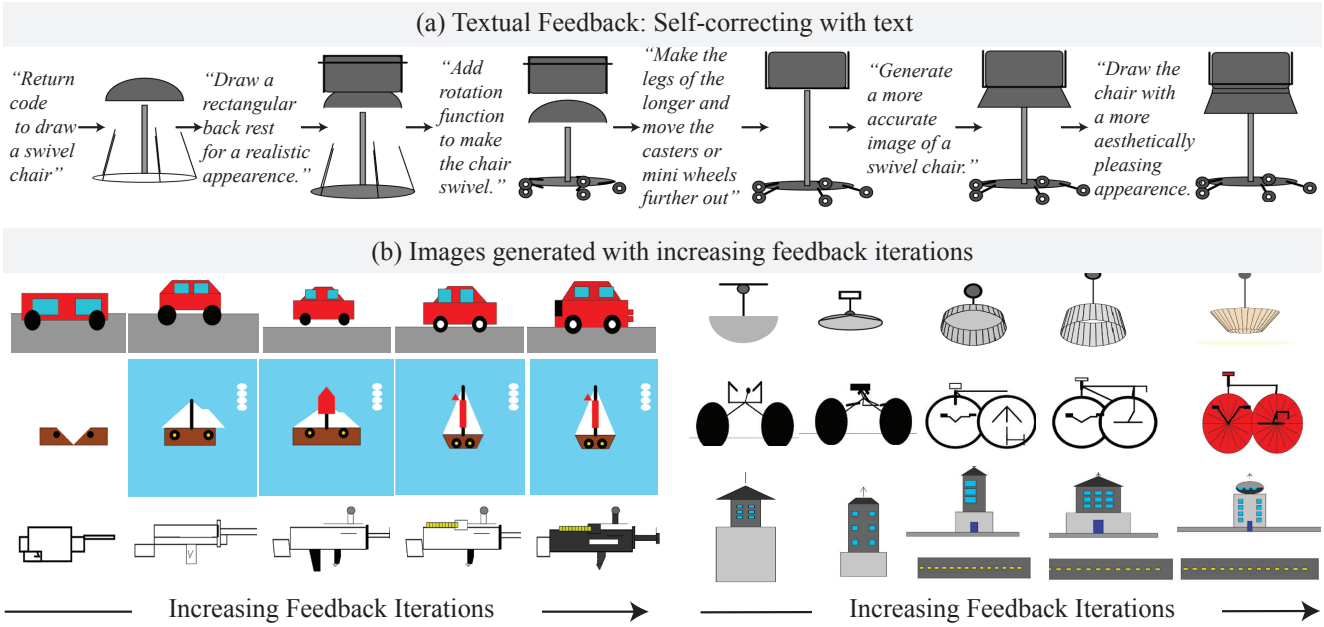


Figure 7. **Improved visual generation with text feedback.** The improvement in the visual generation of models due to feedback is oftentimes gradual, with the addition of a few features at a time over the course of the feedback process. Conditioning on the code generated from the previous iterations, to a limited degree, ensures that the model is constrained to modifying aspects of the current image.

to a particular concept does not necessarily imply its lack of "knowledge" of that particular concept but rather its lack of immediate accessibility. Could direct systematic guidance and textual feedback help improve a model's visual capability? And if so, can this be automated?

Experimental Protocol. The visual competence of a generator language model can be improved by pairing it with itself. This procedure serves as prompt-facilitated "self-talk" and helps scope out the model's internal representation of a particular visual concept over multiple rounds of generation. Additionally, selective step-wise correction of a model's generation gives us insights into whether the model has memorized a fixed "prototype" corresponding to a visual concept or if its representation can be systematically modified by iteratively prompting it to improve its drawing.

Prompt: *"The following [code] does not accurately represent the [concept]. How can you do better?"*.

Evaluation Protocol. To evaluate the improvement in the model's drawing capability, we use the fidelity score that was described in Section 4.1.

Baseline: To assess whether the model's drawing improvements stem from textual feedback or repeated model calls, we conduct a 'variability' baseline. We generate 20 independent outputs for the same concept and report the median CLIP percentile for the best images per concept. Mean variability score is reported in Fig. 8.

The visual competence is improved solely by text-based feedback. We present examples of images generated with iterative feedback in Fig. 7. Visual quality and fidelity to the visual concepts significantly improve with multiple rounds of feedback. Fig. 8 shows that GPT-3.5, a weaker generator than GPT-4, can be made to be nearly as competent of an image generator as GPT-4 with text feedback from itself. Therefore, models can benefit in returning an overall better answer from multiple rounds of self-verification. In fact, with 20 rounds of feedback, the average performance of GPT-3.5 approaches that of GPT-4. Further analysis and visualization for the dataset are in the SM.

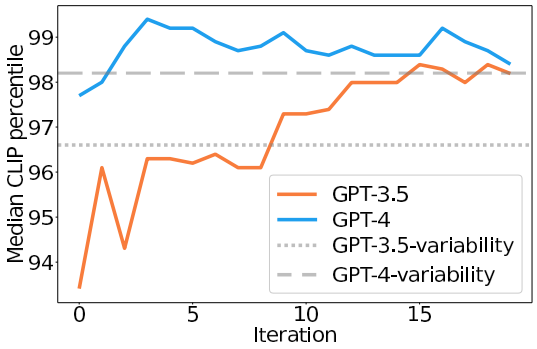


Figure 8. **Text-feedback improves visual generation competence:** Improvement in the quality of the generations is depicted by increasing median CLIP percentile as a function of feedback iterations. The model with textual corrections outperforms selecting the best image from multiple random samples of the same concept.

5. Learning a Vision System from Text

Lastly, we test whether LLM-generated images can be used to train a generally capable vision system for *natural* im-

ages. This relates to abundant existing work [3, 5, 6, 21, 35] that studies the same phenomena by pretraining vision systems using only procedurally generated images.

Training and evaluation protocol. We use unsupervised contrastive learning, which allows pretraining vision backbones with an unlabeled set of images, and follow the training and evaluation protocol of [6]. We train a ResNet-50 [17] using the MoCo-v2 method [11] over a dataset of 1.3M 384×384 images generated by LLMs, for 200 epochs with batch size of 256. For comparison, we generate datasets of the same size and image resolution with 4 different procedural generation methods [5, 6, 21]. After training, we evaluate the performance of models trained on each of the datasets using two approaches: (i) training a linear layer on top of the frozen backbone for ImageNet-1k classification for 100 epochs, and (ii) using 5-nearest neighbor retrieval on ImageNet-100 [38]. The latter is of special interest, as nearest-neighbor retrieval shows that models trained solely on LLM generated data yield powerful representations for natural images, without the need of training a linear layer. This can be qualitatively seen in Fig. 9.

Baselines and datasets. We compare our LLM generated images against existing procedurally generated images. These include simple generative programs like dead-levaves [6], fractals [21], and StyleGAN [6], each consisting of a single program that generates highly diverse images. We also consider the Shaders-21k dataset [5], a large collection of procedural image programs, each one producing a family of images of high diversity. Our LLM-generated dataset consists of all the available images obtained using the different LLMs as described in Sec. 4.1, a total of 80k images. We augment these by randomly sampling convex combinations of six data points using MixUP [44] (shown to be effective for other synthetic images [5]), and reach a total of 1.3M images. As an ablation, we report an additional experiment that excludes GPT-4 images from the training (a 31k subset of the total 80k). Additional dataset breakdowns (*i.e.* each LLM individually) are reported in the SM.

Analysis. Table 2 shows that models trained with only LLM-generated images outperform simple datasets like dead-leaves or fractals, but are still inferior to alternatives. Through visual inspection of the data, we attribute this inferiority to the lack of texture in most LLM-generated images, as observed in the figures throughout the paper and in the SM. To overcome this, we combine the Shaders-21k dataset [5], which generates texture-rich images with the samples obtained from LLMs. This is done by sampling with 50% chance from either dataset and applying MixUP. As seen in Table 2, these models outperform all procedural generation-based alternatives, showing that (i) LLM-generated images combined with textures are powerful representations for natural images, and (ii) scaling up the number of generative image procedures (by combining the procedures in Shaders-21k with those sampled from LLMs) improves overall performance, as predicted in [5]. We also note that the models trained with or without GPT-4 images achieve roughly the same performance. We end this section with the conclusion that LLMs, processing only textual inputs and outputs, can produce images with useful visual properties that complement previous procedural generation approaches for training visual systems.

Figure 9. **Nearest Neighbors Retrieval on ImageNet-100.** Nearest neighbors on ImageNet-100 for a randomly initialized network and a network trained with all our LLM-generated images.

	Pre-training Dataset	I-1k Linear	I-100 5-NN
Random Init.	None	4.36	4.28
Real	Places	55.59	57.04
Procedural	Dead-leaves	20.00	12.76
	FractalDB-1k	23.86	17.24
	StyleGAN	38.12	33.00
	S-21k	44.83	43.24
	LLMs (w/o GPT-4)	33.60	22.42
	LLMs (w/ GPT-4)	36.16	27.44
	LLMs (w/o GPT-4) + S-21k	45.79	**43.40**
	LLMs (w/ GPT-4) + S-21k	**46.03**	43.36

Table 2. **Learning a vision system.** Top-1 Linear evaluation on ImageNet-1k and 5-NN on ImageNet-100, for a ResNet-50 pretrained with different real and procedurally generated datasets including LLM's generated images.

6. Conclusions

We show that LLMs learn visual properties of the real world, and can depict them in the form of procedural image code. We do so by first analyzing the properties of the samples they generate, and showing that the model's competence can be further improved through text feedback from the model itself. Finally, we show that the images produced by these models can be used to train useful vision backbones for downstream tasks on natural images, gracefully complementing alternative approaches.

References

[1] Githubcopilot. https://github.com/features/copilot. 3

[2] Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, Stella Frank, Ellie Pavlick, and Anders Søgaard. Can language models encode perceptual structure without grounding? a case study in color. In *Conference on Computational Natural Language Learning*, 2021. 3

[3] Connor Anderson and Ryan Farrell. Improving fractal pre-training. *CoRR*, abs/2110.03091, 2021. 3, 8

[4] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J Fleet. Synthetic data from diffusion models improves imagenet classification. *arXiv preprint arXiv:2304.08466*, 2023. 3

[5] Manel Baradad, Chun-Fu Chen, Jonas Wulff, Tongzhou Wang, Rogerio Feris, Antonio Torralba, and Phillip Isola. Procedural image programs for representation learning. In *Advances in Neural Information Processing Systems*, 2022. 3, 8

[6] Manel Baradad Jurjo, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. *Advances in Neural Information Processing Systems*, 34:2556–2569, 2021. 3, 8

[7] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12:43–77, 1994. 3

[8] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. 1, 3, 4

[9] Catherine Chen, Kevin Lin, and Dan Klein. Constructing taxonomies from pretrained language models. *arXiv preprint arXiv:2010.12813*, 2020. 1, 3

[10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021. 3

[11] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297, 2020. 8

[12] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1841–1850, 2019. 3

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 5

[14] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 3

[15] Constantin Eichenberg, Sidney Black, Samuel Weinbach, Letitia Parcalabescu, and Anette Frank. Magma–multimodal augmentation of generative models through adapter-based finetuning. *arXiv preprint arXiv:2112.05253*, 2021. 2

[16] Yuling Gu, Bhavana Dalvi Mishra, and Peter Clark. Do language models have coherent mental models of everyday things? *arXiv preprint arXiv:2212.10029*, 2022. 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016. 8

[18] José Hernández-Orallo. Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement. *Artificial Intelligence Review*, 48:397–447, 2017. 2

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 5

[20] Yining Hong, Li Yi, Josh Tenenbaum, Antonio Torralba, and Chuang Gan. Ptr: A benchmark for part-based conceptual, relational, and physical reasoning. *Advances in Neural Information Processing Systems*, 34:17427–17440, 2021. 2

[21] Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without natural images. *CoRR*, abs/2101.08515, 2021. 3, 8

[22] Belinda Z Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021. 1, 3

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 4

[24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 2

[25] Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. Dissociating language and thought in large language models: a cognitive perspective. *arXiv preprint arXiv:2301.06627*, 2023. 2

[26] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pages 4040–4048, 2016. 3

[27] Jack Merullo, Louis Castricato, Carsten Eickhoff, and Ellie Pavlick. Linearly mapping from image to text space. *arXiv preprint arXiv:2209.15162*, 2022. 1, 2

[28] Roma Patel and Ellie Pavlick. Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations*, 2022. 3

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 5

[30] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016. 3

[31] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137:24–37, 2015. 3

[32] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Defossez, Jade Copet, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2023. 3

[33] Mert Bulent Sariyildiz, Karteek Alahari, Diane Larlus, and Yannis Kalantidis. Fake it till you make it: Learning transferable representations from synthetic imagenet clones. In *CVPR 2023–IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 3

[34] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022. 2

[35] Sora Takashima, Ryo Hayamizu, Nakamasa Inoue, Hirokatsu Kataoka, and Rio Yokota. Visual atoms: Pre-training vision transformers with sinusoidal waves. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18579–18588, 2023. 3, 8

[36] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*, 2019. 1, 3

[37] Yoad Tewel, Yoav Shalev, Idan Schwartz, and Lior Wolf. Zero-shot image-to-text generation for visual-semantic arithmetic. *arXiv preprint arXiv:2111.14447*, 2021. 1, 2

[38] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *CoRR*, abs/1906.05849, 2019. 8

[39] Yonglong Tian, Lijie Fan, Phillip Isola, Huiwen Chang, and Dilip Krishnan. Stablerep: Synthetic images from text-to-image models make strong visual representation learners. *arXiv preprint arXiv:2306.00984*, 2023. 3

[40] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021. 2

[41] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022. 3

[42] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023. 2

[43] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023. 2

[44] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. 8

[45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[46] Ziyin Zhang, Chaoyu Chen, Bingchang Liu, Cong Liao, Zi Gong, Hang Yu, Jianguo Li, and Rui Wang. A survey on language models for code, 2023. 3

[47] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 4

[48] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019. 4

[49] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023. 2