Painter: Teaching Auto-regressive Language Models to Draw Sketches

Reza Pourreza[†], Apratim Bhattacharyya[†], Sunny Panchal[‡], Mingu Lee[†], Pulkit Madan[‡], Roland Memisevic[‡]
Qualcomm AI Research[*]
[†]San Diego CA, USA, [‡]Markham ON, Canada
{pourreza,aprabhat,sunnpanc,mingul,pmadan,rmemisev}@qti.qualcomm.com

Abstract

Large language models (LLMs) have made tremendous progress in natural language understanding and they have also been successfully adopted in other domains such as computer vision, robotics, reinforcement learning, etc. In this work, we apply LLMs to image generation tasks by directly generating the virtual brush strokes to paint an image. We present Painter, an LLM that can convert user prompts in text description format to sketches by generating the corresponding brush strokes in an auto-regressive way. We construct Painter based on off-the-shelf LLM that is pre-trained on a large text corpus, by fine-tuning it on the new task while preserving language understanding capabilities. We create a dataset of diverse multi-object sketches paired with textual prompts that covers several object types and tasks. Painter can generate sketches from text descriptions, remove objects from canvas, and detect and classify objects in sketches. Although this is an unprecedented pioneering work in using LLMs for auto-regressive image generation, the results are very encouraging.

1. Introduction

Large language models (LLMs) are growing at an incredible pace and becoming ubiquitous solutions in every domain [4, 6, 1, 3, 28]. This tremendous success is partly owed to the auto-regressive nature of these models *i.e.*, they look at the past and predict the future.

Image generation and text-to-image conversion have seen a fast progress recently [25, 26, 5]. Current methods, despite very impressive results, are not explainable . As such, it is hard to address the shortcomings of these methods.

Here, we present *Painter*, an LLM that is employed for image generation. Unlike the existing image generation methods [25, 5], *Painter* draws sketches the way humans do *i.e.*, by generating a sequence of brush strokes in an auto-regressive way. Since this is an unprecedented work in this domain, we start with a relatively easier task which is sketch generation.

To train such a network, a dataset of text–image pairs is needed where the images should be expressed in the form of brush strokes. The only existing dataset in a similar format is Quick-Draw [16] which is a collection of 50 million class-label–drawing pairs across 345 categories. Since Quick-Draw always has a single object and there are no text descriptions, we create a new dataset by including single or multiple objects in a drawing, defining a composition or a relationship between them, and assigning a text prompt to each sample from a list of tasks. By training *Painter* on the created dataset, it not only is able to draw sketches, it can also perform other tasks like completing incomplete sketches, wiping objects off a drawing, reproducing a sketch by generating the corresponding brush strokes and detecting and classifying the objects in a drawing.

Since the LLM used in *Painter* needs to be multi-modal to consume interleaved text–image data, we make the necessary modifications to the vanilla LLM architecture to convert it to a language-vision model. This is done by adding residual cross-attention blocks that measure cross-attention between image features and hidden states of LLM. Furthermore, we equip the LLM with a visual feedback loop to monitor the state of the canvas as image generation progresses. This is similar to a robotics problem setting where the agent observes the state frequently.

Our contributions are as follows: we introduce a model which to the best of our knowledge is the first to create images using auto-regressive language generation, we create a dataset of text-description–sketch pairs where the sketches are expressed in the format of strokes, and we enhance visual grounding in LLMs via feedback loop, cross-attention layers, and multitasking.

2. *Painter*

In this section, we provide more technical details about the data generation process, model design, and the overall training method.

[*] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

2.1. Dataset

To train *Painter*, we need a dataset of text-description–sketch pairs where the sketches should be in strokes format *i.e.*, all the brush movements should be recorded. To the best of our knowledge, Quick-Draw is the only large-scale dataset of this type. It consists of 50 million class-label–sketch pairs from 345 individual classes. There are two limitations in Quick-Draw for being directly used to train *Painter*: 1) there is a single object in each sample. This could limit the capability of *Painter* to learn the relationships between objects, object counts, and object compositions. 2) The text descriptions in Quick-Draw are limited to class labels. This provides a poor description of the objects in the sample and lacks concepts such as size and location. To alleviate these two limitations, we create the Multi-Object-Quick-Draw dataset. More details are provided in the next two subsections.

2.1.1 Multi-object sketches

Each sketch sample in Multi-Object-Quick-Draw contains one or more objects where the objects come from Quick-Draw and go through some processing. We establish an association between these objects by defining a relationship between them or adding relative location tags to them.

Relationships: to define a relationship between the objects, we follow Sketchforme's [15] approach with some modifications as follows. We initially parse the Visual Genome [21] relationships and select the relationships where both subject and object are present in Quick-Draw classes. Then we perform the followings processing per selected relationship, 1) normalize the bounding boxes of subject and object to our canvas size (256×256) with a small random perturbation, 2) randomly select the subject and the object from Quick-Draw based on their associated classes, 3) scale their strokes to fit the normalized bounding boxes, and 4) put them on the canvas.

Location tags: the process above exhausts a small portion of Quick-Draw. For the remaining Quick-Draw samples, we divide them randomly into groups of 1, 2, 3, or 4 objects from the same or different classes and randomly place them across the canvas after the required size normalization.

2.1.2 Text descriptions

We define task-dependent text descriptions for the sketches in Multi-Object-Quick-Draw *i.e.*, we associate a random task to the sketch from a list of predefined tasks and define a prompt based on the selected task.

Tasks: while the primary application of *Painter* is text-to-sketch conversion, we train it on auxiliary tasks to improve the performance on the primary task via better object, location, and relationship grounding and adding complementary capabilities including wiping out object from a sketch and understanding the contents of a given sketch. Currently, the 6 following tasks are defined to create the dataset, however, this list can grow arbitrarily.

- `generate-all`: includes drawing a single or multiple objects on a blank canvas.
- `generate-partial`: includes completing a partial object or adding new objects to a sketch.
- `remove-all`: includes wiping off the object(s) of a sketch.
- `remove-partial`: includes removing an object from a multi-object sketch.
- `reproduce`: includes reproducing a given sketch by generating the strokes that form the sketch.
- `classification`: includes counting and classifying the objects of a sketch.

Prompt text: For each sketch in Multi-Object-Quick-Draw, we randomly select a task. Then, depending on the task and the number of objects and the associations between the objects in the sketch, we define a prompt text. Table 1 shows the default prompt texts for each task at different scenarios where the scenario can be the number of objects, whether a specific location is defined, whether a relationship exists between the objects, etc. In order to diversify prompt texts, we extend the default prompt texts by rephrasing them in several ways using a pre-trained BLOOM-176B model [27]. We randomly select one of the generated prompts to assign to a sketch.

Once a task-dependent prompt is assigned to a sketch, a prompt sketch and a ground-truth sketch are generated where both could include modifications compared to the original sketch based on the selected task. The prompt sketch is given to the network as a part of the prompt and the ground truth sketch is used for supervised training. Task-dependent modifications can vary widely, for example in the `generate-all` task, all the objects are removed from the sketch to generate the prompt sketch and a blank canvas is given to the network to begin with, while the ground-truth sketch does not include any modifications.

2.2. Model details

This section provides more details about the *Painter* model, including how prompts and responses are encoded, how *Painter* digests interleaved texts and images, and how to equip *Painter* with a visual feedback loop to monitor the state of the canvas.

Prompt format: We use the HTML format to encode the prompts and the responses *i.e.*, commands and responses enclosed between `<command>` and `</command>` tags and `<response>` and `</response>` tags, respectively. We encode strokes in a similar way and include stroke color and thickness as follows:

```
<stroke> color R G B width W points
```

Table 1. Default prompts per task for different scenarios.

Task	Scenario	Default prompt
GENERATE-ALL	SINGLE W/O LOCATION	Draw a sketch of <class-article><class-name>
	SINGLE W/ LOCATION	Draw <class-article><class-name><location-tag> this sketch
	MULTI W/ REL	Draw a sketch of <relationship-tag>
	MULTI W/O REL	Draw a sketch of <objects-list>
GENERATE-PARTIAL	SINGLE	Complete this sketch as <class-article><class-name>
	MULTI W/ LOCATION	Add <class-article><class-name><location-tag> this sketch
	MULTI W/O LOCATION	Add <class-article><class-name> to this sketch
REMOVE-ALL	SINGLE	Remove <class-article><class-name> from this sketch
	MULTI	Remove all the objects from this sketch
REMOVE-PARTIAL	MULTI W/ LOCATION	Remove <class-article><class-name><location-tag> this sketch
	MULTI W/O LOCATION	Remove <class-article><class-name> from this sketch
REPRODUCE	ALL	Reproduce this sketch
CLASSIFICATION	SINGLE	What is the class of this sketch
	MULTI W/ LOCATION	What is the object <location-tag> this sketch
	MULTI W/O LOCATION	What are the objects in this sketch

* <location-tag> ∈ {at the top of, at the bottom of, at the center of, at the right side of, at the left side of, at the top right corner of, at the top left corner of, at the bottom right corner of, at the bottom left corner of}.
* <class-article> ∈ {a, an, the} depending on <class-name>.
* <relationship-tag> includes subject, object, and the relationship between them.
* <objects-list> refers to the objects in the sketch with their counts.
* rel refers to the cases where a relationship (from Visual Genome) exists between the objects.

x_1 y_1, x_2 y_2, ..., x_L y_L </stroke>

where x_i and y_i are the coordinates of the points in a stroke in string format and L is the length of the stroke. Draw actions use black ink and thickness of 1 pixel, while remove actions redraw objects in white ink and thickness of 2 pixels.

To process interleaved texts and images in the model, we insert an image-placeholder in the text wherever an image is needed. As such, the full prompt text for the classification task becomes:

```
<command> What is the class of this
sketch <image-placeholder> </command>
<response> A tree </response>
```

where the response section is used in training only. The same applies to other tasks.

Multi-modal LLM: We use an off-the-shelf pre-trained LLM and modify it similar to [2] to receive and process interleaved texts and images. The overall block diagram is shown in Figure 1. As can be seen there, images are fed to a separate head and are consumed via residual single-head cross-attention components [8, 23] wherever there is a correspond-

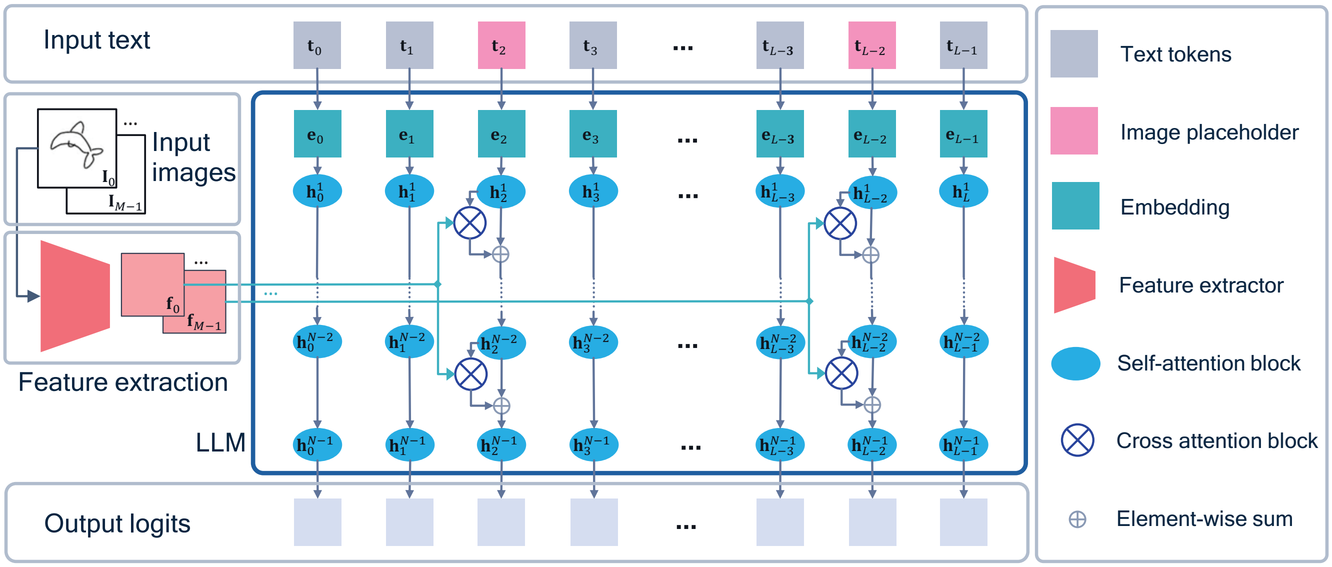


Figure 1. Block diagram of the multi-modal model.

Table 2. Iterative inference steps.

Step	Prompt	Response
1	<command> Draw an apple <image-placeholder> </command>	<response> <stroke> color ... width ... points ... </stroke>
2	<command> Draw an apple <image-placeholder> </command> <response> <stroke> color ... width ... points ... </stroke> <image-placeholder>	<stroke> color ... width ... points ... </stroke>
...	...	...
N-1	<command> Draw an apple <image-placeholder> </command> <response> <stroke> color ... width ... points ... </stroke> <image-placeholder> ... <stroke> color ... width ... points ... </stroke> <image-placeholder>	<stroke> color ... width ... points ... </stroke>
N	<command> Draw an apple <image-placeholder> </command> <response> <stroke> color ... width ... points ... </stroke> <image-placeholder> ... <stroke> color ... width ... points ... </stroke> <image-placeholder> <stroke> color ... width ... points ... </stroke> <image-placeholder>	</response>

** The first <image-placeholder> in prompt corresponds to a blank canvas that is passed to the model to draw on and the subsequent ones correspond to canvas state feedbacks after each stroke.

ing image placeholder in the text. To formalize the cross-attention behavior, let us assume $\mathbb{T} = \{\mathbf{t}_0, \mathbf{t}_1, ..., \mathbf{t}_{L-1}\}$ and $\mathbb{I} = \{\mathbf{I}_0, ..., \mathbf{I}_{M-1}\}$ denote the sequences of text tokens (including the image placeholder token) and images that are fed to the model. Here, L and M represent the lengths of text tokens and image sequences, respectively.

In the text input head of the LLM, the discrete text tokens $\mathbf{t}_i$ are converted to continuous embeddings $\mathbf{e}_i$ and passed through the self-attention blocks of the LLM to generate the hidden states $\mathbf{h}_i^l$, where both $\mathbf{e}$ and $\mathbf{h}$ belong to $\mathbb{R}^{L\times H}$, l and H represent the LLM layer index and the embedding/hidden-state dimensionality. In the image input head of the LLM, the images $\mathbf{I}_i$ are converted to features $\mathbf{f}_i \in \mathbb{R}^{M\times F}$ where F represents image features dimensionality after flattening.

Where there is an image-placeholder in the text, let us assume location j, cross-attention between the corresponding image features $\mathbf{f}_i$ and the hidden states $\mathbf{h}_j^l$ are measured and added to the hidden states. In the cross-attention component, keys and values are extracted from $\mathbf{f}_i$ and queries are extracted from $\mathbf{h}_j^l$. This is done for all the LLM layers except the last one as formulated below:

$$\mathbf{h}_j^l \mathrel{+}= \text{Cross-Attn}(\mathbf{f}_i, \mathbf{h}_j^l), \text{ for } 0 \leq l < L-1 \qquad (1)$$

It is worth noting that we concatenate positional embeddings to image features $\mathbf{f}_i$ before passing them to cross-attention blocks, to preserve spatial information.

Visual feedback: In order to mimic the way humans paint by looking at the canvas while drawing, we equip *Painter* with a visual feedback loop to monitor the state of the canvas while generating strokes. As such, the generated strokes are applied on the canvas on-the-fly using an off-the-shelf line drawer. During training, prompts are augmented with the intermediate canvas states. During inference, recursive prompting is done *i.e.*, once a stroke is generated by the LLM, is it drawn on the canvas, the feedback is applied, and a subsequent prompt with the updated prompt is executed. This is summarized in table 2 for the `generate-all` task.

2.3. Loss

We finetune the LLM used in our model with a standard masked cross-entropy loss function via supervised training. We measure loss in the token domain on the response section only while the image placeholders are masked out. It is worth mentioning that we use the default LLM tokenizer's vocabulary as all the prompt contents are in string format and we do not introduce any special tokens.

3. Experiments

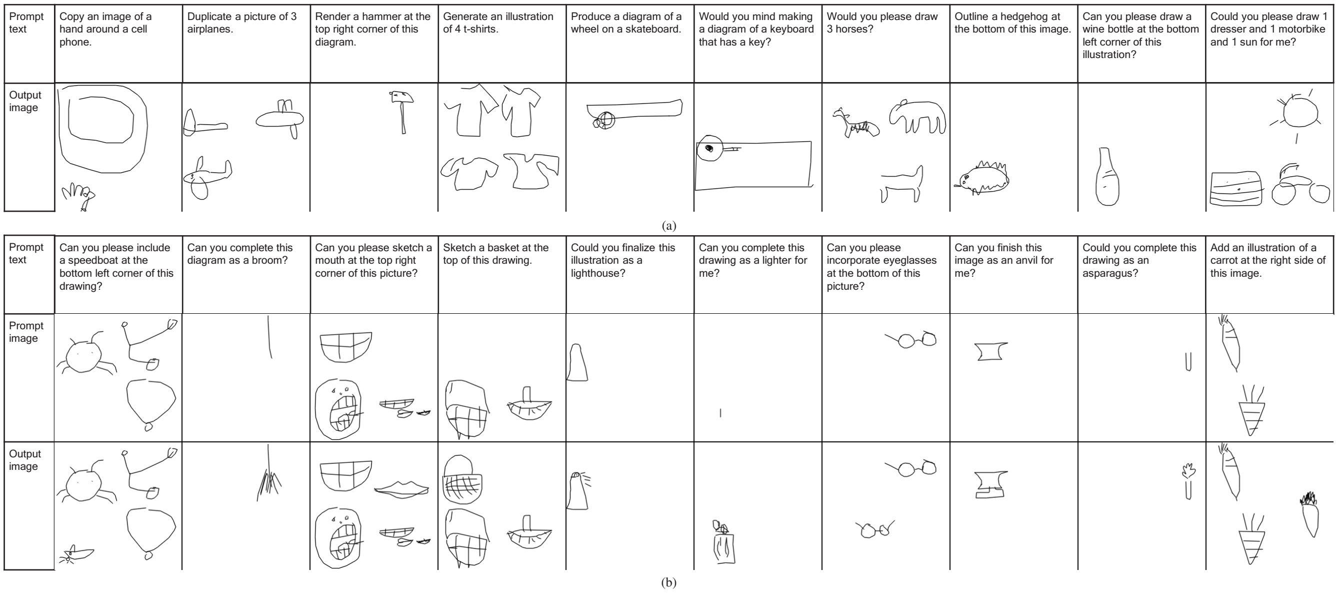
3.1. Training setup

The multi-modal architecture in *Painter* is general enough to work with almost any off-the-shelf LLM. Here, we use two pre-trained LLMs from the OPT family [32], specifically OPT-125M and OPT-1.3B. Most vision-language models that are used in high-level vision tasks such as visual QA and image/video captioning [1, 9], use a pre-trained ViT [7] or a CLIP [22] image encoder to extract visual features. However, these image feature extractors are highly biased toward high-level visual features, while in *Painter*, pixel-

Table 3. Quantitative results.

Model	Train Dataset	`classification`
		`accuracy (%)`
			`remove-all`
			`PSNR (dB)`
				`remove-partial`
				`PSNR (dB)`
					`reproduce`
					`PSNR (dB)`

OPT-125M	MULTI-OBJECT-QUICK-DRAW	34.50	20.15	22.69	17.74
OPT-125M	MULTI-OBJECT-QUICK-DRAW + THE PILE	17.57	20.25	22.95	17.21
OPT-1.3B	MULTI-OBJECT-QUICK-DRAW	3.20	19.56	22.27	17.20
OPT-1.3B	MULTI-OBJECT-QUICK-DRAW + THE PILE	40.26	20.64	23.90	16.97



(a)

Prompt text	Copy an image of a hand around a cell phone.	Duplicate a picture of 3 airplanes.	Render a hammer at the top right corner of this diagram.	Generate an illustration of 4 t-shirts.	Produce a diagram of a wheel on a skateboard.	Would you mind making a diagram of a keyboard that has a key?	Would you please draw 3 horses?	Outline a hedgehog at the bottom of this image.	Can you please draw a wine bottle at the bottom left corner of this illustration?	Could you please draw 1 dresser and 1 motorbike and 1 sun for me?
Output image										

(b)

Prompt text	Can you please include a speedboat at the bottom left corner of this drawing?	Can you complete this diagram as a broom?	Can you please sketch a mouth at the top right corner of this picture?	Sketch a basket at the top of this drawing.	Could you finalize this illustration as a lighthouse?	Can you complete this drawing as a lighter for me?	Can you please incorporate eyeglasses at the bottom of this picture?	Can you finish this image as an anvil for me?	Could you complete this drawing as an asparagus?	Add an illustration of a carrot at the right side of this image.
Prompt image										
Output image										

Figure 2. Selected (a) `generate-all` and (b) `generate-partial` results.

level visual information is required for better grounding and accurate guidance of the network. Hence, we use a pre-trained ResNet-50 [13] for image feature extraction.

During training, the LLM is fine-tuned, the feature extractor remains frozen, and the randomly initialized cross-attention layers are trained. We train the models on Multi-Object-Quick-Draw. Multi-Object-Quick-Draw contains about 20 million samples, but here we use a subset of it with around 2M samples including all the samples with a relationship from Visual Genome (around 300K samples) and 1.7M samples with location tags. We use a small portion of this dataset containing around 1000 samples for evaluation and reserve the rest for training and validation. Since the LLM is already pre-trained on a large text corpus and is highly skilled in natural language understanding, we regularize the training on The Pile [10] dataset to retain the natural language understanding capabilities of the LLM.

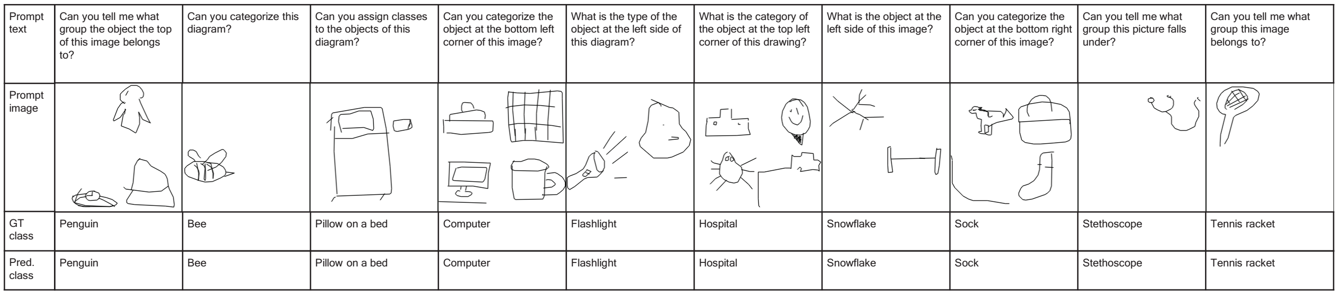
We train the models using Adam [19] optimizer with a learning rate of $1e-5$ on two A100 Nvidia cards for one training epoch with a batch size of 4.

During inference, we use greedy sampling in the `classification` task and use top-p sampling with $p = 0.9$ for the other tasks.

3.2. Results

We can evaluate the tasks that *Painter* is trained on as follows:

- `remove-all`, `remove-partial`, and `reproduce` can be evaluated using pixel-level metrics such as `MSE` and `PSNR`, since exact results are expected from them.
- `classification` accuracy can be measured by comparing the detected classes with the ground-truth classes.
- `generate-all` and `generate-partial` are not trivial for quantitative evaluation and need a user-study to



Prompt text	Can you tell me what group the object the top of this image belongs to?	Can you categorize this diagram?	Can you assign classes to the objects of this diagram?	Can you categorize the object at the bottom left corner of this image?	What is the type of the object at the left side of this diagram?	What is the category of the object at the top left corner of this drawing?	What is the object at the left side of this image?	Can you categorize the object at the bottom right corner of this image?	Can you tell me what group this picture falls under?	Can you tell me what group this image belongs to?
Prompt image										
GT class	Penguin	Bee	Pillow on a bed	Computer	Flashlight	Hospital	Snowflake	Sock	Stethoscope	Tennis racket
Pred. class	Penguin	Bee	Pillow on a bed	Computer	Flashlight	Hospital	Snowflake	Sock	Stethoscope	Tennis racket

Figure 3. Selected `classification` results.

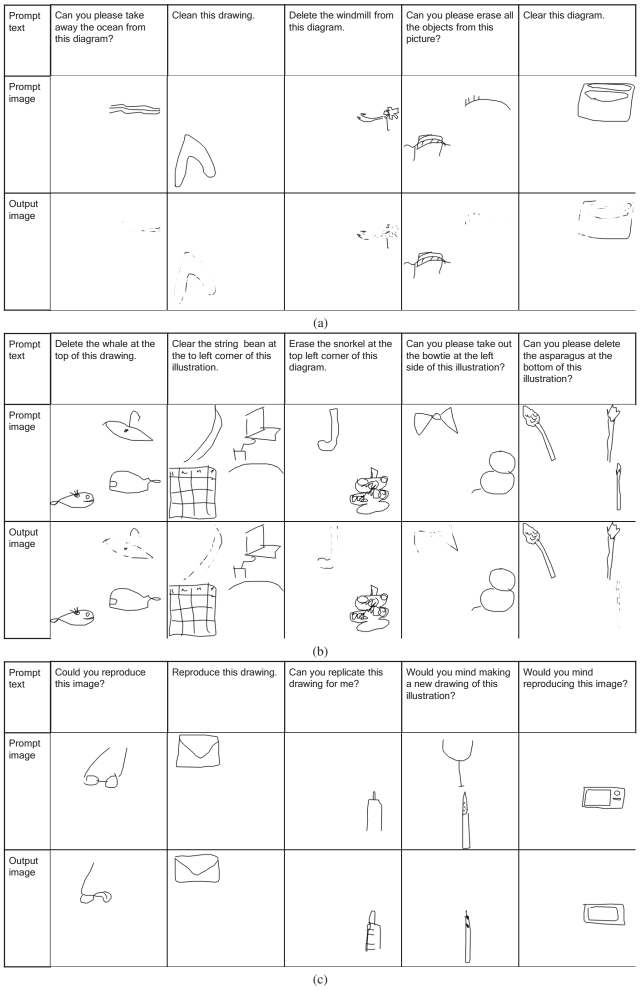

Prompt text
Can you please take away the ocean from this diagram?
Clean this drawing.
Delete the windmill from this diagram.
Can you please erase all the objects from this picture?
Clear this diagram.
Prompt image
Output image
(a)
Prompt text
Delete the whale at the top of this drawing.
Clear the string bean at the to left corner of this illustration.
Erase the snorkel at the top left corner of this diagram.
Can you please take out the bowtie at the left side of this illustration?
Can you please delete the asparagus at the bottom of this illustration?
Prompt image
Output image
(b)
Prompt text
Could you reproduce this image?
Reproduce this drawing.
Can you replicate this drawing for me?
Would you mind making a new drawing of this illustration?
Would you mind reproducing this image?
Prompt image
Output image
(c)

Figure 4. Selected (a) `remove-all`, (b) `remove-partial`, and (c) `reproduce` results.

rate the generated results. Since an extensive user-study is out of budget of this work, for these two tasks we show qualitative results only.

We report quantitative results for `classification` in terms of classification accuracy and for `remove-all`, `remove-partial`, and `reproduce` tasks in terms of `PSNR` in table 3.

We show selected qualitative results for the `generate-all` and `generate-partial` tasks in figure 2. The qualitative results that are shown in this paper are all generated using the *Painter* model based on OPT-1.3B which is trained on Multi-Object-Quick-Draw and regularized on The Pile, unless otherwise noted. As can be seen from this figure, the model understands concepts such as shapes, relevant locations among objects, objects counts, and relationships.

Additionally, we show selected qualitative results for the `classification` task in figure 3 and for the `remove-all`, `remove-partial`, and `reproduce` tasks in figure 4.

3.3. Discussion

As can be noted in table 3, the classification accuracy is quite low. Our investigation leads to two main causes for this. The first reason is that we count a classification label as correct if there is an exact match between the ground truth label and the label that *Painter* generates. There are examples such as ”pillow under a cat” versus ”cat on a pillow” that are conceptually the same, but are counted as wrong in our evaluation. The second reason is that there are objects classes that are very similar visually and are very hard to distinguish even for a human rater, such as ”birthday cake” versus ”cake” or ”pen” versus ”marker”. Some examples from both categories of reasons are shown in figure 5.

To examine the attention maps that *Painter* learns in the cross-attention blocks, we visualize them for a few `classification` examples in figure 6. As can be seen here, *Painter* can locate the objects in the input images and correctly attend to them, this is more pronounced in the first and third cross-attention blocks.

One of the shortcomings that *Painter* has in its current form, is the limited number of object categories that it can identify which is limited to the 345 classes in Quick-Draw. We are working on expanding the object vocabulary in *Painter* via techniques like reinforcement learning.

4. Related work

In spite of the great progress of auto-regressive LLMs and their extensive footprints in every domain, their full potential is not released in the image generation domain, as the latest methods utilize their representation learning aspect but do not benefit from their powerful auto-regressive nature. Therefore, we group existing image generation works into two broad categories: non-auto-regressive and auto-regressive methods.

4.1. Non-auto-regressive methods

Variational autoencoders (VAEs) [20, 29] are one of the pioneering works for image generation. Later, Generative Adversarial Models (GANs) [11, 18, 17] improved a lot upon VAEs on image generation and were considered the best-performing method, until recently. Diffusion models and their variants [14, 25, 26, 5] are now state-of-the-art in this domain as they are very creative and generate mesmerizing results.

4.2. Auto-regressive methods

There are very a limited number of works in this category and to the best of our knowledge, none has a pre-trained auto-regressive LLM backbone for image generation. Sketch-RNN [12] is an early work that generates sketches from a

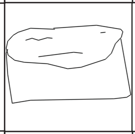
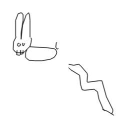
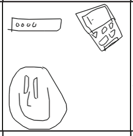
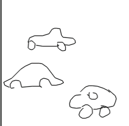
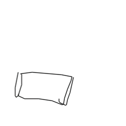
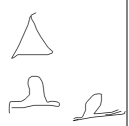
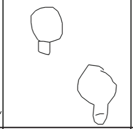
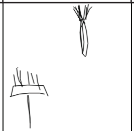
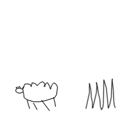
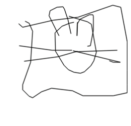
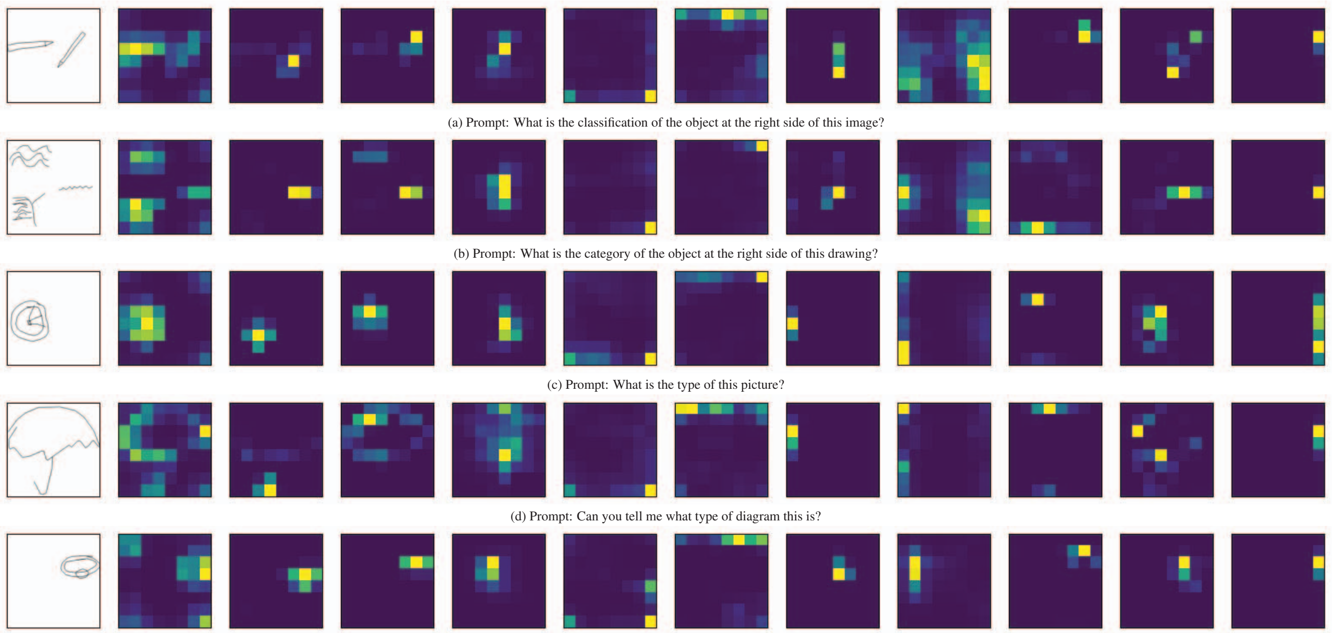
Prompt text	Can you identify the type of this illustration?	What is the type of the object at the bottom right corner of this image?	Can you tell me what group the object at the bottom of this illustration belong to?	What is the class of the object at the top of this illustration?	What is the class of this drawing?	Can you determine the category of the object at the top of the drawing?	Can you assign a class to the object the top left corner of this illustration?	Can you assign a class to the object at the top right corner of this illustration?	What are the categories of the object of this illustration?	What are the objects in this picture?
Prompt image										
GT class	Hot tub	Lightning	Face	Car	Pillow	Hat	Light bulb	Paintbrush	Grass by a sheep	Pillow under a cat
Pred. class	Postcard	Knee	Smiley face	Police car	Diving board	Triangle	Hot air balloon	Carrot	Sheep on grass	Cat on bed

Figure 5. Selected incorrect `classification` results.

single class using a recurrent VAE. Sketchforme [15] extends to multi-object drawing via a combination of a transformer and a Sketch-RNN. Sketchformer [24] can classify, retrieve, and reconstruct sketches via an embedding that is learned in stroke space. Parti [31] is a hybrid approach that uses an encoder-decoder LLM for language understanding and embedding and a pre-trained ViT-VQGAN [30] for image generation.

Conclusions

We present *Painter*, the first-ever LLM-based image generation solution that draws sketches by generating strokes in an auto-regressive way. We build the Multi-Object-Quick-Draw dataset consisting of diverse text-description–sketch pairs where the sketches contain single or multiple objects with relationships or relative location tags between the objects, and the text descriptions are devised from a list of pre-defined tasks with additional prompt diversification using a very large language model. We modify the LLM architecture by adding residual cross-attention layers to make it a vision-language model, and additionally add a visual feedback loop to actively observe the state of the canvas. Our results show the viability of *Painter*'s approach. There are shortcomings in *Painter* including the limited number of object categories that will be addressed in future work.



(a) Prompt: What is the classification of the object at the right side of this image?

(b) Prompt: What is the category of the object at the right side of this drawing?

(c) Prompt: What is the type of this picture?

(d) Prompt: Can you tell me what type of diagram this is?

(e) Prompt: Can you tell me the category of this drawing?

Figure 6. Cross-attention maps of an OPT-125M-based *Painter* model for several `classification` examples. There are 11 attention maps per example that from left to right correspond to the first layer to the eleventh layers in OPT-125M (there all cross-attention blocks in all 12 OPT-125M layers except the last one).

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[2] Apratim Bhattacharyya, Sunny Panchal, Mingu Lee, Reza Pourreza, Pulkit Madan, and Roland Memisevic. Look, remember and reason: Visual reasoning with grounded rationales. *CoRR*, abs/2306.17778, 2023.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. *CoRR*, abs/2212.06817, 2022.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[5] Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, José Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. *CoRR*, abs/2301.00704, 2023.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311, 2022.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[8] Zi-Yi Dou, Aishwarya Kamath, Zhe Gan, Pengchuan Zhang, Jianfeng Wang, Linjie Li, Zicheng Liu, Ce Liu, Yann LeCun, Nanyun Peng, Jianfeng Gao, and Lijuan Wang. Coarse-to-fine vision-language pre-training with fusion in the backbone. In *NeurIPS*, 2022.

[9] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. *CoRR*, abs/2303.03378, 2023.

[10] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[12] David Ha and Douglas Eck. A neural representation of sketch drawings. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[15] Forrest Huang and John F. Canny. Sketchforme: Composing sketched scenes from text descriptions for interactive applications. In François Guimbretière, Michael S. Bernstein, and Katharina Reinecke, editors, *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology, UIST 2019, New Orleans, LA, USA, October 20-23, 2019*, pages 209–220. ACM, 2019.

[16] Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. The quick,draw! - a.i. experiment. 2016.

[17] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 852–863, 2021.

[18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4401–4410. Computer Vision Foundation / IEEE, 2019.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[21] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vis.*, 123(1):32–73, 2017.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[23] Tanzila Rahman, Hsin-Ying Lee, Jian Ren, Sergey Tulyakov, Shweta Mahajan, and Leonid Sigal. Make-a-story: Visual memory conditioned consistent story generation. In *CVPR*, 2023.

[24] Leo Sampaio Ferraz Ribeiro, Tu Bui, John P. Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 14141–14150. Computer Vision Foundation / IEEE, 2020.

[25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022.

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

[27] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100, 2022.

[28] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761, 2023.

[29] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017.

[30] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[31] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *CoRR*, abs/2206.10789, 2022.

[32] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer.

OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068, 2022.