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.
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:

\[ <stroke> \text{color } R \text{ G } B \text{ width } W \text{ points} \]
\[ x_1, y_1, x_2, y_2, \ldots, x_L, y_L \langle /stroke \]

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
<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-

**Table 1. Default prompts per task for different scenarios.**

<table>
<thead>
<tr>
<th>Task</th>
<th>Scenario</th>
<th>Default prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GENERATE-ALL</strong></td>
<td>SINGLE W/O LOCATION</td>
<td>Draw a sketch of ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt; )</td>
</tr>
<tr>
<td></td>
<td>SINGLE W/ LOCATION</td>
<td>Draw ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt;&lt;\text{location-tag}&gt; ) this sketch</td>
</tr>
<tr>
<td></td>
<td>MULTI W/ REL</td>
<td>Draw a sketch of ( &lt;\text{relationship-tag}&gt; )</td>
</tr>
<tr>
<td></td>
<td>MULTI W/O REL</td>
<td>Draw a sketch of ( &lt;\text{objects-list}&gt; )</td>
</tr>
<tr>
<td><strong>GENERATE-PARTIAL</strong></td>
<td>SINGLE</td>
<td>Complete this sketch as ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt; )</td>
</tr>
<tr>
<td></td>
<td>MULTI W/ LOCATION</td>
<td>Add ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt;&lt;\text{location-tag}&gt; ) this sketch</td>
</tr>
<tr>
<td></td>
<td>MULTI W/O LOCATION</td>
<td>Add ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt; ) to this sketch</td>
</tr>
<tr>
<td><strong>REMOVE-ALL</strong></td>
<td>SINGLE</td>
<td>Remove ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt; ) from this sketch</td>
</tr>
<tr>
<td></td>
<td>MULTI</td>
<td>Remove all the objects from this sketch</td>
</tr>
<tr>
<td><strong>REMOVE-PARTIAL</strong></td>
<td>MULTI W/ LOCATION</td>
<td>Remove ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt;&lt;\text{location-tag}&gt; ) this sketch</td>
</tr>
<tr>
<td></td>
<td>MULTI W/O LOCATION</td>
<td>Remove ( &lt;\text{class-article}&gt;&lt;\text{class-name}&gt; ) from this sketch</td>
</tr>
<tr>
<td><strong>REPRODUCE</strong></td>
<td>ALL</td>
<td>Reproduce this sketch</td>
</tr>
<tr>
<td><strong>CLASSIFICATION</strong></td>
<td>SINGLE</td>
<td>What is the class of this sketch</td>
</tr>
<tr>
<td></td>
<td>MULTI W/ LOCATION</td>
<td>What is the object ( &lt;\text{location-tag}&gt; ) this sketch</td>
</tr>
<tr>
<td></td>
<td>MULTI W/O LOCATION</td>
<td>What are the objects in this sketch</td>
</tr>
</tbody>
</table>

* \( <\text{location-tag}> \in \{ \text{at the top of, at the bottom of, at the center of, at the right side of, at the left side of, at the top right corner of, at the top left corner of, at the bottom right corner of, at the bottom left corner of} \} \)

* \( <\text{class-article}> \in \{ \text{a,a,an, the} \} \) depending on \( <\text{class-name}> \)

* \( <\text{relationship-tag}> \) includes subject, object, and the relationship between them.

* \( <\text{objects-list}> \) 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.
ing image placeholder in the text. To formalize the cross-
attention behavior, let us assume \( T = \{t_0, t_1, \ldots, t_{L-1}\} \) and \( I = \{I_0, \ldots, I_{M-1}\} \) denote the sequences of text tokens (includ-
ing 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 \( t_i \) are converted to continuous embeddings \( e_i \) and passed through the self-attention blocks of the LLM to generate the hidden states \( h^i_1 \), where both \( e \) and \( h \) belong to \( \mathbb{R}^{L \times H} \) and \( l \) and \( H \) represent the LLM layer index and the embedding/state dimensionality. In the image input head of the LLM, the images \( I_i \) are converted to features \( f_i \in \mathbb{R}^{M \times F} \) 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 \( f_i \) and the hidden states \( h^j_l \) are measured and added to the hidden states. In the cross-attention component, keys and values are extracted from \( f_i \) and queries are extracted from \( h^j_l \). This is done for all the LLM layers except the last one as formulated below:

\[
h^j_{l+} = \text{Cross-Attn}(f_i, h^j_l), \quad 0 \leq l < L - 1 \tag{1}
\]

It is worth noting that we concatenate positional embeddings to image features \( f_i \) 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, it is 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Train Dataset</th>
<th>classification accuracy (%)</th>
<th>remove-all PSNR (dB)</th>
<th>remove-partial PSNR (dB)</th>
<th>reproduce PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT-125M</td>
<td>MULTI-OBJECT-QUICK-DRAW</td>
<td>34.50</td>
<td>20.15</td>
<td>22.69</td>
<td>17.74</td>
</tr>
<tr>
<td>OPT-125M</td>
<td>MULTI-OBJECT-QUICK-DRAW + THE PILE</td>
<td>17.57</td>
<td>20.25</td>
<td>22.95</td>
<td>17.21</td>
</tr>
<tr>
<td>OPT-1.3B</td>
<td>MULTI-OBJECT-QUICK-DRAW</td>
<td>3.20</td>
<td>19.56</td>
<td>22.27</td>
<td>17.20</td>
</tr>
<tr>
<td>OPT-1.3B</td>
<td>MULTI-OBJECT-QUICK-DRAW + THE PILE</td>
<td>40.26</td>
<td>20.64</td>
<td>23.90</td>
<td>16.97</td>
</tr>
</tbody>
</table>
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 [17] dataset to retain the natural language understanding capabilities of the LLM.

We train the models using Adam [19] optimizer with a learning rate of $1 \times 10^{-5}$ 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 $\text{MSE}$ and $\text{PSNR}$, 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
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 QuickDraw. 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
<table>
<thead>
<tr>
<th>Prompt</th>
<th>Can you identify the type of this illustration?</th>
<th>What is the type of the object at the bottom right corner of this image?</th>
<th>Can you tell me what group the object at the bottom of this illustration belong to?</th>
<th>What is the class of the object at the top of this illustration?</th>
<th>Can you determine the category of this object at the left of this drawing?</th>
<th>Can you assign a class to the object the top left corner of this illustration?</th>
<th>Can you assign a class to the object the top right corner of this illustration?</th>
<th>What are the categories of the object of this illustration?</th>
<th>What are the objects in this picture?</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT Class</td>
<td>Hot tub</td>
<td>Lightning</td>
<td>Face</td>
<td>Car</td>
<td>Pillow</td>
<td>Hat</td>
<td>Light bulb</td>
<td>Paintbrush</td>
<td>Greens by a sheep</td>
</tr>
<tr>
<td>Pred class</td>
<td>Postcard</td>
<td>Knee</td>
<td>Smiley face</td>
<td>Police car</td>
<td>Diving board</td>
<td>Triangle</td>
<td>Hot air balloon</td>
<td>Carrot</td>
<td>Sleep on grass</td>
</tr>
</tbody>
</table>

Figure 5. Selected incorrect classification results.

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.

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


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