# Drawing with LLMs

Build and submit Kaggle Packages capable of generating SVG images of specific concepts

**header**
ファイル · 61 KB



## Overview

Given a text prompt describing an image, your task is to generate Scalable Vector Graphics (SVG) code that renders it as an image as closely as possible. Your submissions will be built with Kaggle Packages, a new feature for building reusable models as Python libraries.

We have two Starter Notebooks with more details:

1. Official Starter Notebook that outlines our new Kaggle Packages feature and demos a trivial model.
2. Getting Started with Gemma 2 Notebook that demos a more advanced model but assumes familiarity with Kaggle Packages.

**Start**
2 months ago

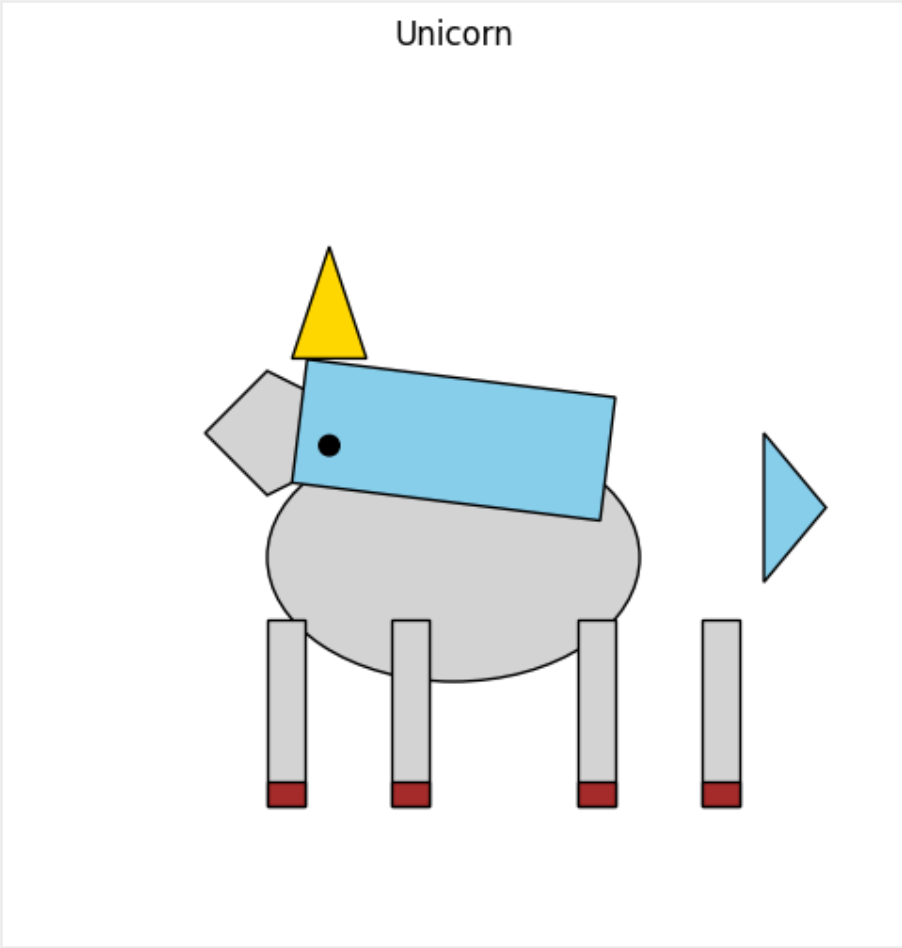
**Close**
a month to go

Merger & Entry

## Description

## linkkeyboard_arrow_up

Specialized solutions can significantly outperform even the impressive capabilities of generative models like ChatGPT and Gemini while providing greater transparency into how they're built. And while LLMs may demonstrate "sparks of AGI", their capacity to generate image-rendering code is one area that needs improvement.




This competition challenges you to build practical, reusable solutions for image generation that follow robust software engineering patterns. Given a text description of an image, your task is to generate SVG code which renders it as closely as possible. Scalable Vector Graphics (SVG) is a vector image format that uses XML to describe two-dimensional graphics which can be scaled in size without quality loss.

Your submission, created using the new Kaggle Packages feature, will be a class Model with a predict() function which generates SVG code for a given prompt. The end results will be a set of deployable model

packages evaluated on their ability to deeply reason about abstract descriptions and translate them into precise, executable code.
If you have feedback or questions, please let us know in this competition's Discussion forum. We appreciate your input as we improve and continue to develop Kaggle Packages as a new way to run Kaggle Competitions.

## Package Requirements

## linkkeyboard_arrow_up

### This is a Package Competition

This is the first competition that requires submissions to be Kaggle Packages. This new feature expands on experimental functionality introduced in a few recent competitions and allows you to:

- Submit models defined with a predict() function for scoring
- Create simpler solutions as Kaggle's scoring infrastructure will handle the test set iteration
- Share and discover solutions that are reusable as installable inferenceable models with defined dependencies using kagglehub

When you submit your Package to this competition, we will install your Notebook's Package in a Container with your Notebook's Accelerator and Environment settings, with Internet disabled. Our scoring system will call your Model to run inference over the hidden test set and determine your score.
To learn how to create a Kaggle Package to submit to this competition, fork one of our two starter notebooks (basic or advanced) and check out our documentation.

### Package Competitions are Code Competitions

Submissions to this competition must be made through Notebooks. In order for the "Submit" button to be active after a commit, the following conditions must be met:

- CPU Notebook <= 9 hours run-time
- GPU Notebook <= 9 hours run-time
- Internet access disabled
- Freely & publicly available external data is allowed, including pre-trained models

Please see the Code Competition FAQ for more information on how to

submit. And review the code debugging doc if you are encountering submission errors.

## Evaluation

## linkkeyboard_arrow_up

Submissions are evaluated by the **SVG Image Fidelity Score** between the given descriptions and the submitted SVG code. For each given description in the test set, your model must produce an SVG image depicting the scene described.

The evaluation proceeds through several stages.

First, in order to ensure the generated images adhere to the spirit of the competition, we check that the SVG code satisfies a number of constraints:

1. No SVG may more than 10,000 bytes long.
2. Each SVG may only include elements and attributes from an allowlist. Note that CSS style elements are not allowed.
3. No SVG may include any rasterized image data or data from external sources.

Second, we convert each SVG to a PNG image with the cairosvg Python library. To this PNG image we apply a number of preprocessing filters to prevent noise attacks and encourage faithfulness at the level of overall image composition.

Third, a VQA task is applied to the processed PNG with a PaliGemma model. To each description, there are four image-specific questions with either yes-or-no or multiple choice answers. The questions are meant to check how well the rendered image conforms to specific parts of the given description. See TIFA: Text-to-Image Faithfulness Evaluation with Question Answering for more details on the method.

Fourth, we apply OCR text detection and score adjustment. We also use PaliGemma's OCR capabilities to read any text out of a (lightly processed) PNG image. If more than four characters are detected, an exponential penalty is applied to the overall score for that image. The four "free" characters are meant to give some buffer against hallucinated text.The overall VQA score is the product of the OCR penalty and the TIFA task score.

Fourth, we compute a CLIP-based Aesthetic Score. See the CLIP Aesthetic Predictor repo for details.

The final overall score is the harmonic mean of the VQA score and the Aesthetic score, with a (\beta=2) weight favoring the VQA score.

In addition, the evaluation system requires that:

1. Your model returns an SVG within 5 minutes of being passed a description.

2. All SVGs are generated in under 9 hours.

You may review the implementation of the metric here: **SVG Image Fidelity**. This metric imports the svg_constraints package defined here: **SVG Constraints**. You may wish to use this constraints package to help ensure your submissions are valid.

### Timeline

### linkkeyboard_arrow_up

- **February 25, 2025** - Start Date.
- **May 19, 2025** - Entry Deadline. You must accept the competition rules before this date in order to compete.
- **May 19, 2025** - Team Merger Deadline. This is the last day participants may join or merge teams.
- **May 27, 2025** - Final Submission Deadline.

All deadlines are at 11:59 PM UTC on the corresponding day unless otherwise noted. The competition organizers reserve the right to update the contest timeline if they deem it necessary.

### Citation

### linkkeyboard_arrow_up

Ryan Holbrook, DJ Sterling, Paul Mooney, Maggie Demkin, and Elizabeth Park. Drawing with LLMs. https://kaggle.com/competitions/drawing-with-llms, 2025. Kaggle.

## Dataset Description

The competition test data comprises about 500 descriptions of everyday objects and scenes from a variety of domains. Your goal is to create a model that generates SVG depictions of these descriptions.

The descriptions have the following properties:

- The descriptions are of common, generic subjects. No brand name or trademark or personal name occurs in any description. No people, even in generic form, occur in any description.
- The subjects described span about a dozen categories. Three of these categories, landscapes, abstract, and fashion, are present in each of the training, public test, and private test sets. The remaining categories in the public and private sets are unique to each set. More than half of the descriptions come from the three shared categories.
- No description has more than 200 characters. The average

length is around 50 characters.

## Files

- **train.csv** - A set of representative descriptions within the landscapes, abstract, and fashion categories.
- **kaggle_evaluation/** - A Python package with competition helpers, such as the test(Model) function that simulates the evaluation system using a mock test.csv.
- **kaggle_evaluation/test.csv** - An example set of test data to help you author your submissions. When your submission is scored, this placeholder test data will be replaced with the actual test data.

## Evaluation

The dataset contains a Python package in kaggle_evaluation/ with competition helpers such as a test(Model) function. The test.csv file is used by this Python package in Kaggle's scoring system. Kaggle's scoring model reads this file and uses it as inputs to calling your predict() function for generating submissions on the evaluation set.