System Design Document - Drawing with LLMs

Workshop No. 2 — Kaggle Systems Design

Daniel Alonso Chavarro Chapatecua - 20241020066
Carlos Andrés Brito Guerrero - 20241020147
Jose Fernando Ramirez Ortiz - 20241020080

May 10th, 2025

1 Review of Workshop #1 Findings

1.1 System Overview

In Workshop #1, we analyzed a Kaggle competition focused on using Large Language Models (LLMs) to generate images in SVG format from text descriptions. The competition requires designing a system that can interpret textual prompts and produce high-quality, constraint-compliant SVG code that accurately represents the described image.

1.2 Key Insights from Analysis

- **Input Constraints:** The system receives text descriptions limited to 200 characters, averaging around 50 characters, covering various categories including landscapes, abstract art, and fashion.
- **Output Requirements:** The system must generate SVG code under 10,000 bytes, using only permitted SVG elements and attributes, without CSS styling, rasterized image data, or external fonts.
- **Evaluation Metrics:** Submissions are judged on the SVG Image Fidelity Score, measuring how well SVG images match textual descriptions, with penalties for OCR-detected text.
- **System Sensitivity:** High sensitivity was identified in both the Description Preprocessing component and the LLM Model itself.
- **Chaos Factors:** Initial randomness in LLM outputs that stabilizes through continuous learning and feedback loops.

2 System Requirements

2.1 Functional Requirements

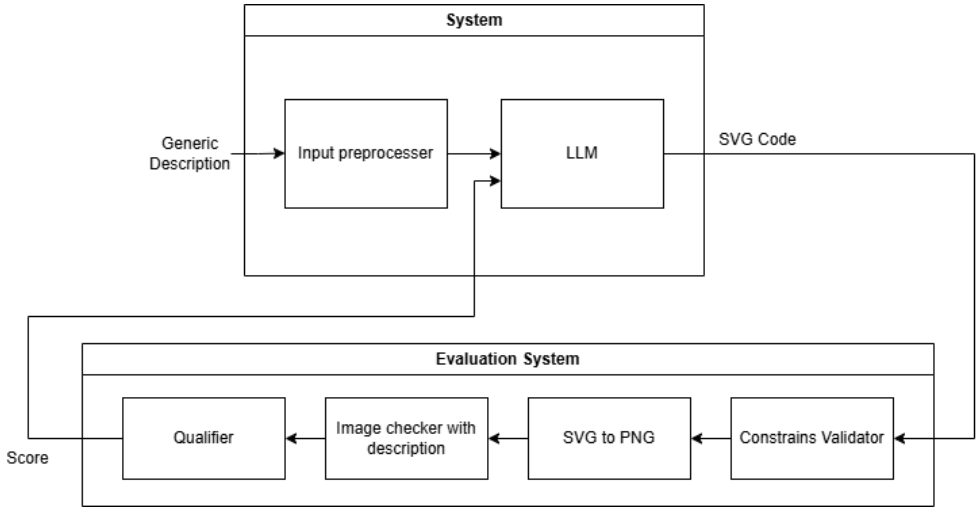
1. The system must accept text descriptions of images (up to 200 characters).
2. The system must preprocess descriptions into standardized prompts for the LLM.
3. The system must generate syntactically correct and constraint-compliant SVG code.
4. The system must optimize SVG output to score highly on the SVG Image Fidelity Score.
5. The system must ensure SVG outputs are under 10,000 bytes.
6. The system must support the generation of images across multiple categories (landscapes, abstract, fashion, etc.).

2.2 Non-Functional Requirements

1. **Performance:** Generate SVG responses within a reasonable timeframe (under 30 seconds per prompt).
2. **Reliability:** Maintain consistent quality across different description categories.
3. **Maintainability:** Design a modular architecture that allows for component replacement or enhancement.
4. **Adaptability:** Support the incorporation of feedback to improve generation quality over time.
5. **Efficiency:** Optimize memory usage during SVG generation to operate within Kaggle's notebook constraints.
6. **Cost Management:** Due to the limited resources, the system must be cheap or even using open source technologies.

3 High-Level Architecture

3.1 Architectural Diagram



3.2 Component Descriptions

3.2.1 Input Handler

- Responsible for receiving and parsing the input text descriptions.
- Performs initial validation and standardization of inputs.
- Acts as the entry point for the system workflow.

3.2.2 Prompt Engineer

- Transforms raw descriptions into structured prompts for the LLM.
- Adds specific instructions about SVG constraints and requirements.
- Applies templating and formatting to optimize LLM understanding.

3.2.3 LLM Generator

- Core component that interprets engineered prompts.
- Generates initial SVG code based on prompt instructions.
- Applies learned patterns from training data and feedback loops.

3.2.4 Evaluation Feedback Loop

- Collects scoring and performance metrics from the evaluation system.
- Provides structured feedback to improve generation quality.
- Informs adjustments to LLM Generator.

3.3 Systems Engineering Principles Applied

- **Modularity:** Each component has a specific responsibility.
- **Separation of Concerns:** Clear division between input handling, prompt engineering, and generation.
- **Feedback Integration:** Explicit feedback loop to support continuous improvement.
- **Interface Standardization:** Well-defined interfaces for seamless integration.

4 Addressing Sensitivity and Chaos

4.1 Prompt Engineering Strategy for Sensitivity

1. Standardized Prompt Templates.
2. Prompt Versioning.
3. Instruction Clarity.
4. Constraint Embedding.

4.2 LLM Optimization for Sensitivity

1. Few-Shot Learning.
2. Temperature Control.
3. Multiple Generation Strategy.
4. Category-Specific Fine-Tuning.

4.3 Chaos Mitigation Strategies

1. Progressive Learning.
2. Evaluation Weighting.
3. Error Pattern Recognition.

4.4 Monitoring and Error Handling

- Error Classification System.
- Graceful Degradation.
- Exception Handling.

5 Technical Stack and Implementation

5.1 Recommended Technologies

5.1.1 Core Technologies

- **Programming Language:** Python 3.9+
- **LLM Framework:** Transformers (Hugging Face models)
- **SVG Processing:** `lxml` and `svglib`
- **Vector Operations:** NumPy

5.1.2 LLM Models

- Base Model: DeepSeek, Gemma, Gemmini 2.5 Flash, Qwen models or Free tier tunned models.

5.1.3 Development and Testing

- Jupyter Notebooks
- Git
- `pytest` or Kaggle testing package

5.2 Implementation Plan

5.2.1 Phase 1: Core Infrastructure

- Set up model loading and inference pipeline.
- Implement basic prompt engineering templates.

5.2.2 Phase 2: Optimization and Refinement

- Fine-tune models on category-specific training data.
- Develop feedback collection and integration mechanisms.

5.2.3 Phase 3: Quality Assurance and Scaling

- Comprehensive testing.
- Performance optimization.
- Documentation and preparation.

5.3 Design Patterns

- **Factory Pattern:** For generating category-specific prompts.
- **Strategy Pattern:** For different SVG optimization techniques.
- **Observer Pattern:** For monitoring performance and metrics.

6 Conclusion

This system design document outlines a comprehensive approach to the "Drawing with LLMs" Kaggle competition. By adopting a modular architecture with clear feedback loops and implementing strategies to handle high-sensitivity components, we aim to create a robust system for generating SVG images from text descriptions. The design balances performance with constraints, focusing on continuous improvement and adaptability.

7 References

1. Drawing with LLMs. Kaggle. https://www.kaggle.com/competitions/drawing-with-llms
2. Systems Analysis & Design Workshop #1 (April 13, 2025)
3. Hugging Face Transformers Documentation. https://huggingface.co/docs/transformers/
4. SVG 1.1 Specification. https://www.w3.org/TR/SVG11/
5. Parameter-Efficient Fine-Tuning (PEFT). https://huggingface.co/docs/peft/