Benchmarking Open-Source Inpainting Solutions for Magic Replace

Julien Delavande

July 2025

Abstract

The rise of generative AI has enabled powerful new capabilities in image editing and synthesis. In this work, we benchmark a set of state-of-the-art inpainting models available from the open-source community. Our goal is to identify a production-ready solution that can be integrated into Presti's "Magic Replace" functionality, which allows users to edit parts of an image using a brush and a prompt. We evaluate six models on a small, realistic dataset of indoor scenes. Both quantitative (latency, memory usage) and qualitative (visual quality, prompt adherence) metrics are considered. Based on the results, we identify the most robust and deployable solution and provide suggestions for future improvements.

1 Introduction

Inpainting refers to the task of modifying or reconstructing specific regions of an image, either by removing unwanted elements or by adding new ones. With the development of diffusion models and transformer-based architectures, inpainting has seen significant progress and now enables high-quality edits based on text prompts and masks.

Presti's "Magic Replace" feature enables users to modify an image by brushing over an area and describing what they want to see in that region. The challenge lies in selecting a robust and high-quality model that can handle a variety of edits while being efficient and easily deployable.

In this report, we benchmark several inpainting models from the open-source ecosystem and evaluate their performance on realistic prompts. Our main goals are:

- To identify inpainting models that can be programmatically integrated into Presti's app;
- To compare them in terms of inference speed, memory usage, and visual quality;
- To select the most "production-ready" option based on these criteria.

We also propose a set of recommendations for deployment and improvement based on our findings.

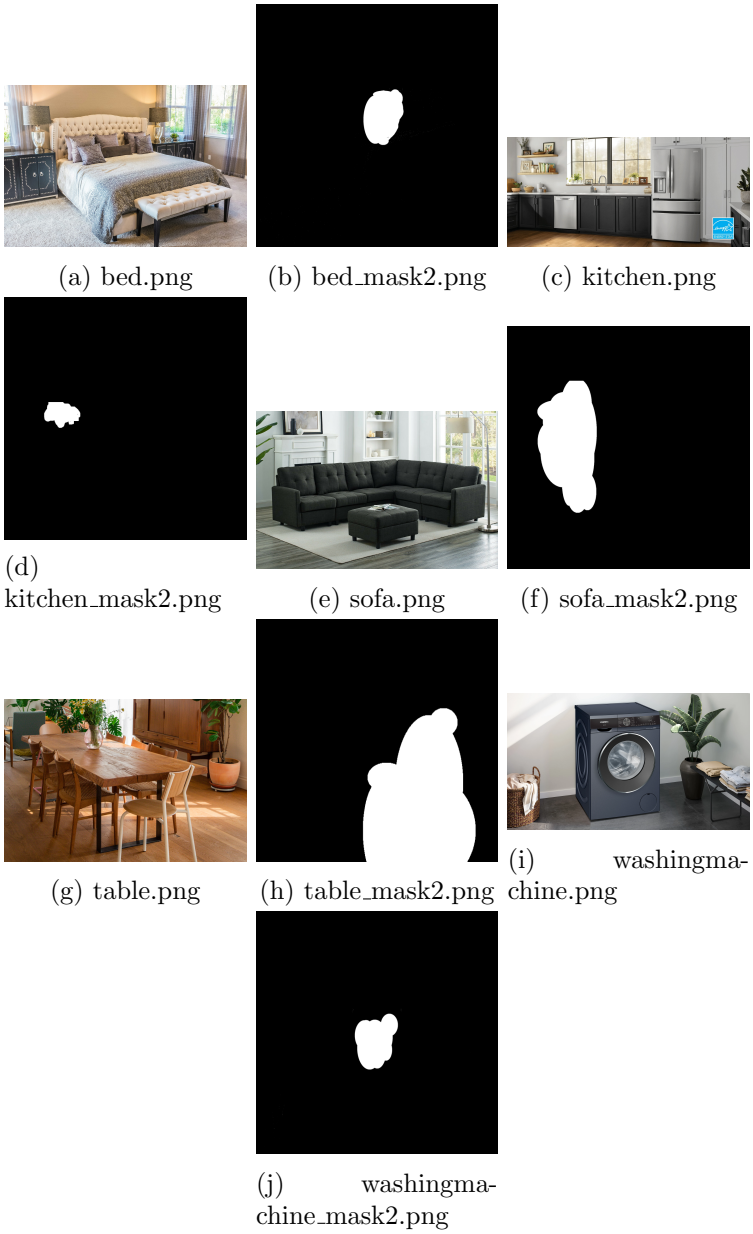
2 Dataset and Evaluation Setup

2.1 Mini-Dataset for Indoor Inpainting

To evaluate the models, we created a small benchmark dataset of 5 indoor scenes inspired by typical user requests on Presti's platform. Each image is accompanied by a mask defining the region to edit and a textual prompt describing the desired modification.

Image	Mask	Prompt
bed.png	bed_mask2.png	*add a dog lying on the bed*
kitchen.png	kitchen_mask2.png	*add a pie on the kitchen counter*
sofa.png	sofa_mask2.png	*add a person sitting on the sofa*
table.png	table_mask2.png	*remove the front chair from the scene*
washingmachine.png	washingmachine_mask2.png	*put some clothes inside the washing machine*

Table 1: Mini-dataset used for inpainting evaluation.



(a) bed.png (b) bed_mask2.png (c) kitchen.png

(d) kitchen_mask2.png (e) sofa.png (f) sofa_mask2.png

(g) table.png (h) table_mask2.png (i) washingmachine.png

(j) washingmachine_mask2.png

Figure 1: Reference images and masks for the 5 prompts. (images are all resized-the mask is the size of the resized image

2.2 Models Evaluated

We selected 6 inpainting models from the Hugging Face Hub. These models were chosen for their popularity, technical diversity, and applicability to Presti's pipeline. Each entry includes

a hyperlink to its Hugging Face page and details on its license.

- `fluxcontrolbeta` — link, licensed under **FLUX.1 [dev] Non-Commercial License**.
- `fluxkontextdev` — link, licensed under **FLUX.1 [dev] Non-Commercial License**.
- `stablediffusionv2 inpainting` — link, licensed under **CreativeML Open RAIL++-M License**.
- `stablediffusion inpainting` (v1.5) link, licensed under **CreativeML Open RAIL++-M License**.
- `stablediffusionxl inpainting` — link, licensed under **CreativeML Open RAIL++-M License**.
- `kandinsky` — link, licensed under **Apache-2.0 License (fully open source)**.

All models except `fluxkontextdev` require a triplet `(image, mask, prompt)` as input. `fluxkontextdev` performs inpainting without a mask, relying solely on the input image and prompt.

Note on licenses: Only `kandinsky` (Apache-2.0) and the `stablediffusion` family (v1.5, v2, XL — under CreativeML Open RAIL++-M) allow for **commercial use**, provided you comply with the license terms. `fluxcontrolbeta` and `fluxkontextdev` are restricted to **non-commercial research or personal use only**.

Commercial Licensing Options for FLUX Models

While the `fluxcontrolbeta` and `fluxkontextdev` models are only licensed for non-commercial research and personal use under the **FLUX.1 [dev] Non-Commercial License**, commercial use is possible via paid plans offered by the developers.

For self-hosted deployments, the **FLUX.1 Kontext [dev]** model can be licensed commercially for:

- **$999/month** for up to **100,000 images**.
- Above this threshold, an additional cost of **$0.01 per image** applies.

Alternatively, FLUX also provides API access to more advanced commercial models:

- **FLUX.1 Kontext [pro]** — `/v1/flux-kontext-pro` — at **$0.04 per image**. This model supports high-speed editing, character consistency, and both regional and full-scene transformations.
- **FLUX.1 Kontext [max]** — `/v1/flux-kontext-max` — at **$0.08 per image**. This premium model delivers top-tier typography generation, high fidelity, and prompt adherence.

These options offer scalable and flexible paths for integrating FLUX technology into commercial products. However, they may require significant GPU resources and financial commitment, especially at scale.

2.3 Evaluation Protocol

We evaluate each model on the 5 prompts using the following metrics:

- **Latency**: average inference time (in seconds) on a single GPU.
- **CUDA Memory**: peak GPU memory usage during generation (in MB).
- **Visual Quality**: subjective analysis of how well the model adhered to the prompt, realism of generation, and image consistency.

Each run was executed with default model parameters unless otherwise specified. Prompts and masks were not modified to adapt to each model, to simulate real deployment scenarios.

3 Quantitative Results

3.1 Inference Time and Memory Usage

To evaluate the practicality of each model for real-world use, we measured both the **average generation time** (latency) and the **peak CUDA memory usage** for each model on our benchmark set.

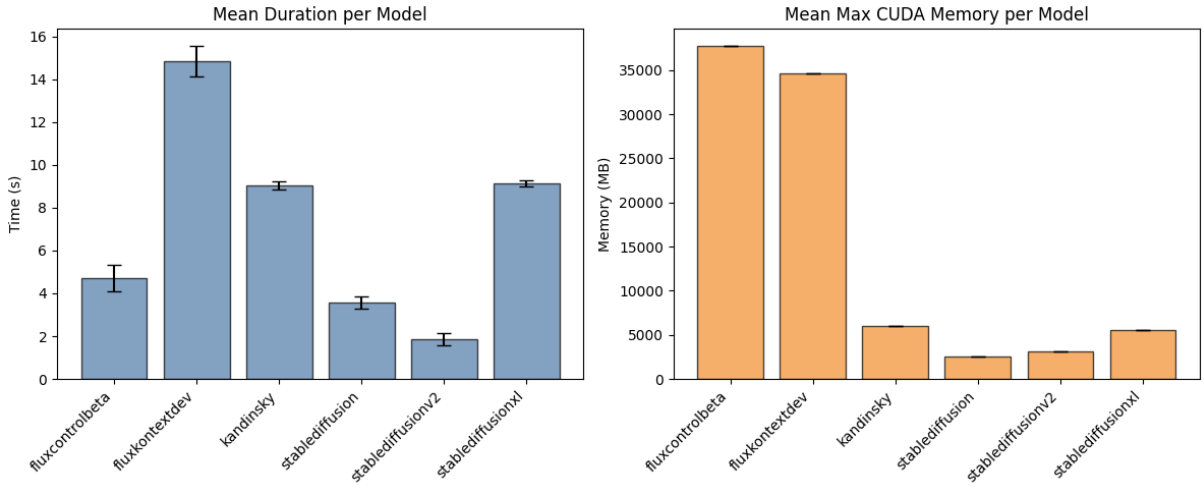


Figure 2: Comparison of mean latency (in seconds) and peak CUDA memory usage (in GB) for each model.

3.2 Observations

The most time-consuming model is `fluxkontextdev`, followed by `kandinsky` and `stablediffusionxl`, which showed similar inference times. On the other hand, `stablediffusionv2` was the fastest model in the benchmark.

Regarding memory usage:

- `fluxcontrolbeta` consumed the most memory, peaking at **37 GB**.
- `fluxkontextdev` followed closely with **35 GB**.
- All other models stayed well below 0.5 GB, with:
  - `stablediffusionv1` using as little as **0.25 GB**,
  - and `stablediffusionv2`, `sdxl`, and `kandinsky` staying under **0.5 GB**.

3.3 Hardware Requirements

These differences in memory usage have strong implications for deployment:

- Models such as `stablediffusionv1`, `v2`, and `kandinsky` can run on consumer-grade GPUs like the NVIDIA RTX 3060 (12 GB) or even Google Colab free tier (T4, 16 GB).
- `fluxcontrolbeta` and `fluxkontextdev` require professional-grade GPUs such as the A100 (40–80 GB), H100, or high-end RTX 6000 Ada, as they exceed 30 GB of peak memory usage during inference.

This trade-off between realism and system requirements will be further discussed in Section 5.

4 Qualitative Results

In addition to latency and memory usage, we performed a qualitative evaluation of the models based on visual inspection. The key criteria were:

- **Prompt adherence**: does the model follow the instruction accurately?
- **Visual realism**: are the generated objects coherent with the image context?
- **Image consistency**: does the inpainting preserve the scene structure and lighting?

4.1 Visual Comparison

We show below a composite grid summarizing the results of all models on all 5 prompts. Each row corresponds to a prompt, with the reference image and the outputs from the six models.

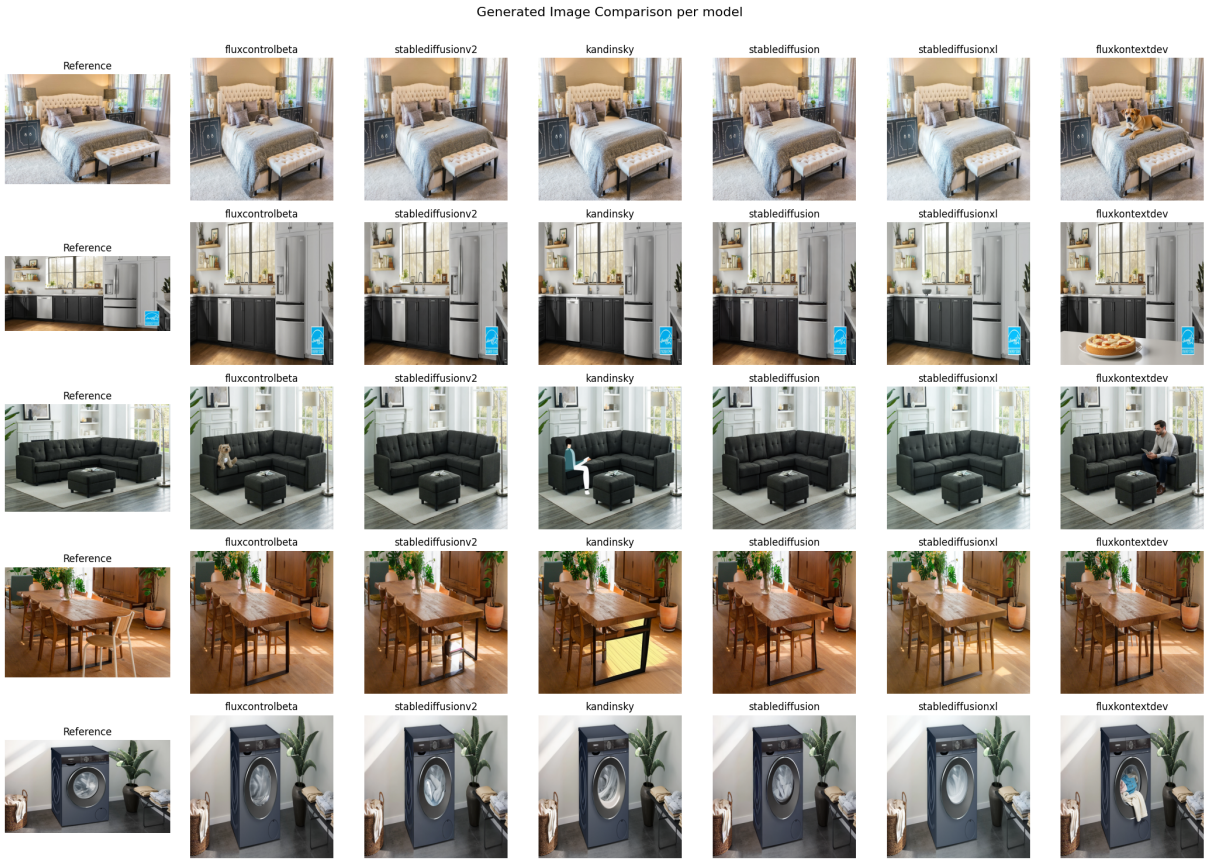


Figure 3: Qualitative comparison across models. Each row corresponds to a different prompt (see Table 1), with results from: FluxControlBeta, StableDiffusion v2, Kandinsky, StableDiffusion v1, StableDiffusionXL, and FluxKontextDev.

4.2 Prompt-by-Prompt Observations

- **Dog on the bed**: Only `fluxcontrolbeta` and `fluxkontextdev` successfully generate a dog. The dog from `fluxkontextdev` is slightly oversized but visually more realistic.
- **Pie on the kitchen counter**: `stablediffusionv2` and `sdxl` generate a bowl instead of a pie. `stablediffusionv1` outputs a mirror-like object. `fluxcontrolbeta` fails completely. `fluxkontextdev` produces a table extension with a large pie—visually coherent.
- **Person on the sofa**: Only `fluxkontextdev` generates a convincing human figure sitting on the sofa. `kandinsky` produces a cartoon-like person, and `fluxcontrolbeta` mistakenly adds a dog.
- **Remove the front chair**: All models succeed to some extent. The most seamless removals are from `fluxcontrolbeta` and `fluxkontextdev`, which reconstruct the background more naturally.
- **Clothes in the washing machine**: `stablediffusion` (v1, v2, XL) and `kandinsky` manage to place clothes inside the drum. `fluxcontrolbeta` fails to modify the scene. `fluxkontextdev` removes the washing machine door entirely and places clothes outside, making the result unrealistic.

4.3 Analysis

`fluxkontextdev` stands out in terms of prompt adherence and realism, especially in complex edits involving object addition. However, its occasional tendency to over-edit (e.g., removing structures like the washing machine door) suggests limited controllability.

On simpler edits such as object removal, several models perform adequately, but `fluxcontrolbeta` and `fluxkontextdev` consistently offer the most convincing reconstructions.

5 Discussion

5.1 Strengths and Weaknesses of Each Model

We summarize below the main trade-offs observed between quality, latency, and resource usage for each model:

Model	Latency	Memory	Quality	Prompt Adherence	Remarks
`fluxcontrolbeta`	Moderate	**Very high (37 GB)**	Medium	Medium	Effective for object removal. Often fails on addition.
`stablediffusionv1`	**Very fast**	**Very low (0.25 GB)**	Low	Low	Produces irrelevant content. Not recommended.
`stablediffusionv2`	**Fastest**	Low	Medium	Medium	Balanced option, but inconsistent.
`stablediffusionxl`	Slow	Low	Medium+	Low-Medium	More detailed images, but lacks alignment with prompts.
`kandinsky`	Slow	Low	Medium	Low-Medium	Artistic output, less realistic.
`fluxkontextdev`	**Slowest**	**Very high (35 GB)**	**High**	**High**	Best overall quality, but overediting and heavy requirements.

Table 2: Summary of trade-offs across all models tested.

5.2 Production-Readiness Analysis

From a deployment standpoint, a good production model must balance:

- Consistency and realism of outputs;
- Efficiency (latency, memory);
- Ease of integration (input format, dependencies).

`fluxkontextdev` clearly provides the best visual quality and prompt adherence. It does not require masks, which simplifies user experience and system integration. However, it is also the most resource-demanding model, requiring more than 30 GB of GPU RAM and producing latency unsuitable for real-time use without batching or model distillation.

`stablediffusionv2` is an efficient and lightweight alternative, suitable for fast edits or fallback scenarios. Its outputs are less impressive, but it could complement heavier models in a hybrid architecture.

5.3 Limitations of the Study

- **Small dataset:** only 5 prompts were used, which limits statistical conclusions.
- **Subjective evaluation:** while objective metrics like latency and memory are precise, visual quality assessments are qualitative and may be biased.
- **Default parameters:** all models were used with their default settings; no prompt tuning or hyperparameter optimization was applied.

5.4 Recommendations

- **Adopt `fluxkontextdev` as the core inpainting model**, especially for cases involving object addition.
- **Use lightweight models like `sd-v2` as a fallback**, or for low-end devices or fast preview modes.
- **Collect and annotate more real user data** from Presti to fine-tune models and better evaluate edge cases.
- **Experiment with prompt engineering or controlnet guidance** to improve accuracy and reduce over-editing.
- **Explore model distillation or quantization** techniques to bring `fluxkontextdev` closer to production latency and footprint.

6 Conclusion

In this report, we conducted a comparative benchmark of six open-source inpainting models suitable for integration into Presti's "Magic Replace" functionality. Our evaluation covered both quantitative metrics—latency and GPU memory usage—and qualitative performance on realistic prompts.

Among the models tested, `fluxkontextdev` emerged as the most promising in terms of visual realism and prompt adherence. Its ability to operate without a mask simplifies the pipeline and improves usability. However, its high latency and memory footprint currently limit its suitability for real-time deployment on modest hardware.

Conversely, lighter models such as `stablediffusionv2` offer a better performance-efficiency trade-off, albeit at the cost of occasional inaccuracies or artifacts.

Based on our findings, we recommend a hybrid deployment strategy: use `fluxkontextdev` for high-precision final renders, and fallback to faster models like `sd-v2` for previews or low-spec devices. Further work should focus on collecting more user-centered data, optimizing prompts, and compressing heavy models through distillation or quantization.

Ultimately, as the generative AI ecosystem continues to evolve, production-level inpainting capabilities are becoming increasingly accessible—opening exciting prospects for platforms like Presti to deliver rich, intuitive, and controllable image editing experiences.