Agentic Retrieval and Editing System for Image Generation

Berwyn Berwyn
Stanford University
berwyn@stanford.edu

Abstract

Agentic Retrieval and Editing System for Image Generation (ARES-GEN) aims to bridge natural language queries with visual content by developing an image editing pipeline that retrieves images from the database and edit them when necessary. We integrate CLIP for the purpose of semantic search, BLIP for image captioning, GPT-4 for discrepancy list construction, and InstructPix2Pix for text-guided image editing. We evaluate the method on a subset of 1000 images from Flickr Image Dataset. While the retrieval component performs exceptionally well, the image editing quality is limited by the use of lightweight InstructPix2Pix that can be run locally, which leads to the editing result successfully capture the semantic intent but with reduced resolution as compared to the SoTA models.

1. Introduction

The ability to convert natural language descriptions into a realistic image is one of the most well-known problem in computer vision and natural language processing. Even though it is easy for human to describe what they want to see and mentally modify existing images, creating automated computational systems that could perform similar task remains a technical challenge. The translation from natural language intent to visual representation has important applications for content creation, fashion, digital media manipulation, and human-computer interaction.

Most approaches to convert natural language queries to images rely on generating the images from scratch using difussion models or GAN [4]. Even though these methods can produce good and realistic results, they often struggle with generating photorealistic content and maintaining fine-grained details, especially when dealing with complex scenes. These diffusion-based generated images often exhibit artifacts, inconsistent lighting, and unrealistic textures which makes them look synthetic.

Our project ARES-GEN stems from a key idea. Instead of generating images entirely from scratch pixel per pixel, we can leverage the rick and realistic information contained in existing photographs and apply modifications to match user queries better. By starting with real image that already captures rich realistic features, we can preserve the photorealistic qualities which could be difficult to generate.

1.1. Input and Output Specification

The input to our pipeline is a natural language query which describes the desired image (e.g, "A man with a black shirt dunking a basketball"). Our system then uses a semantic search to find a semantically similar image. In this case, the model might retrieve an image of a man with a white shirt dunking a basketball. The pipeline then detected the discrepancy in the shirt color and performed the necessary modification. The final output of the pipeline is the edited image that hopefully matches the user query.

This approach has practical applications in content creation, digital art generation, rapid prototyping, and accessible image editing tools. By automating the translation from textual descriptions into visual modifications, we hope to democratize image editing capabilities.

2. Related Work

. The pipeline built on this project involves multiple research domains: text-to-image generation, image editing, and vision-language retrieval systems. We organize the related work into four categories: text-to-image generation models, instruction-based image editing, image retrieval systems, and hybrid approaches.

2.1. Text-to-Image Generation Models

Many approaches in the text-to-field generation niche use a direct generation method via diffusion models and GANs [4]. Imagen [14] presents a text-to-image diffusion model with photorealism and deep language understanding, representing the SoTA in this category. Similarly, Stable Diffusion uses a latent diffusion approach that focuses on accessibility, which makes widely adopted open-source solutions.

DALL-E 2 [13] also represents a commercial breakthrough that achieved good photorealism. However, relying only on text might not cater to the complex requirements

in various applications. Despite their remarkable success, they still sometimes struggle with fine-grained control and maintaining photorealistic quality in complex scenes.

2.2. Instruction-Based Image Editing

Researchers have also done works related to image editing. InstructPix2Pix proposes a method for editing images from human instructions. Other notable works in this category involve DragGAN [10], which enables point-based manipulation, and Paint-by-Example [17] which uses exemplar-based editing. However, these methods are usually limited by the quality of the diffusion models they use and might introduce artifacts during the editing process.

2.3. Vision-Language Models and Image Retrieval

The popular method for bridging the gap between textual and visual semantic representations is through vision-language models like CLIP [12]. CLIP is a neural network trained on around 400 million text-image pairs using contrastive learning. It uses a vision encoder for images and text encoder for languages and project both into a shared embedding space, which allows direct comparison through cosine similarity.

CLIP enables image retrieval systems where an image with similar vector as the text query could be retrieved. Extensions like ALIGN [8] and FLAVA [15] have improved multimodal understanding. These VLMs help users search over image repositories by finding photos that match their natural language queries.

2.4. Hybrid Retrieval and Generation Approaches

Some works have recently begun to explore the combinations of retrieval and generation. Composed Image Retrieval approaches retrieve images while integrating modifications, though focuses more on retrieval rather than editing. Make-A-Scene [3] combines scene layout with text generation, while GLIDE [9] uses classifier-free guidance for better text conditioning. However, these approaches either focus purely on generation from scratch or perform editing without utilizing large-scale retrieval.

2.5. State-of-the-Art and Current Limitations

The current state-of-the-art for text-to-image generation tasks is dominated by difussion models like DALL-E 3, Midjourney v6, and Stable Diffusion XL. For image editing, InstructPix2Pix becomes the leading approach for instruction-based modification, although it suffers from resolution limitations. Most practitioners still rely on manual image editing tools like photoshop for more control, particularly in professional contexts. Our approach differs from existing work by combining the strength of retrieval for photorealism with instruction-based editing.

3. Methods

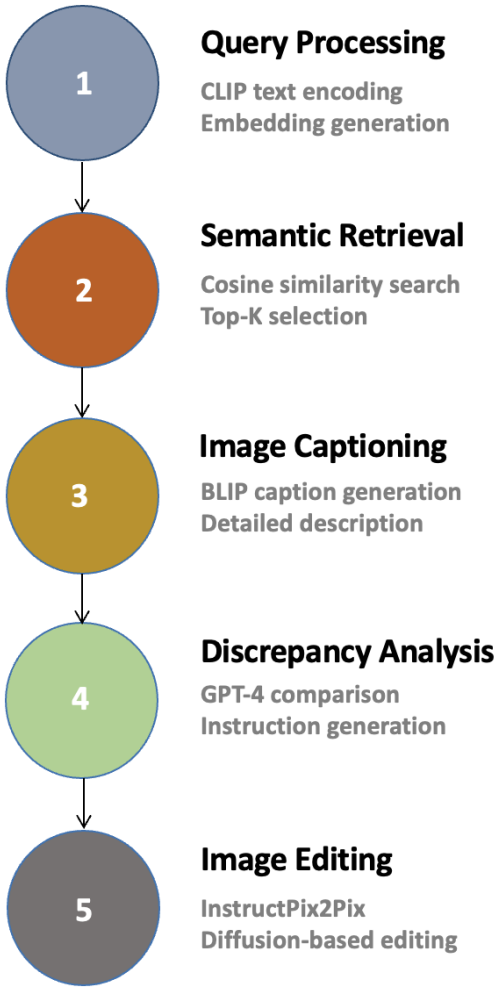

Figure 1. ARES-GEN Workflow

3.1. Query Processing

User first gives an input to the ARES-GEN model in a form of natural language query. The query is then processed with python transformer library's CLIPModel class to produce the text embedding

3.2. Semantic Image Retrieval

The text embedding attained from the previous step is then used to search over the vector database. The search is done by computing the cosine similarity between the text embedding and every image embedding that we have precomputed within the vector database. We retrieved the top-K images with the greatest similarity scores and passed every image to the captioning process that will be explained on the next section.

3.3. Image Captioning with BLIP

During this step, we utilized python transformer library's BLIPForConditionalGeneration class to generate captions for all K images retrieved from the previous step. BLIP model utilized a unified Multimodal Mixture of Encoder-Decoder architecture with ViT as the backbone. The model lies in its boostrapping training paradigm which combines a captioner and a filter that removes noisy web-scraped text. We use this model due to its capability to capture fine-grained visual attributes while still making it feasible to run locally on a CPU. The hyperparameters that we used to initialize the model are:

1. max_new_tokens: this limits the number of new tokens the model can generate in response to the prompt.

2. num_beams: this enables beam search with multiple beams. This means that the model explores multiple possible sequences at every step, keeps the top multiple most likely ones, and continues. This mechanism improves fluency and coherence. More beams tends to give better output but slower inference time.

3. length_penalty: this adjusts how the model scores long versus short sequences during beam search. Values greater than 1.0 penalize shorter sequences while values less that 1.0 penalize longer sequences.

4. no_repeat_ngram_size: this prevents the model from repeating n-gram sequences. For example, without this hyperparameter, one might get an output like "A cat is on the mat. The cat is on the mat.". no_repeat_ngram_size ensures that there is no such redundancy during inference.

3.4. Discrepancy Analysis

One of the core components of our ARES-AGEN model is the discrepancy analysis component, which identifies differences between user queries and retrieved image description. These discrepancies could be differences in attributes, missing objects, and misplaced objects. To analyze discrepancies, we use OpenAI GPT-4 API using this prompt:

You are a helpful assistant that analyzes discrepancies between a user request and an image caption, and suggests precise image edits to match the request.

Your job is to return a structured list of **clear and actionable** edits required to modify the image described by the caption so that it satisfies the user request.

Each edit must have:
- 'edit_type': a short category like 'color', 'object', 'action', 'setting', or 'attribute'.
- 'description': a **specific instruction** phrased like a command, e.g., "Change the man's shirt to black." or "Make the man dunk the basketball."

Do not mention what's already correct — only list what must be **added, changed, or removed**.

If there are no required edits, return an empty list.
Input:
Image caption: "retrieved_caption"
User request: "user_query"

Respond with a JSON array of suggested edits ["edit_type": "action", "description": "Make the man dunk the basketball." , "edit_type": "color", "description": "Change the man's shirt to black."]

3.4.1 OpenAI Structured Outputs

We also used OpenAI structured outputs to force the model to output in a form of list of edit instructions. We compute the discrepancies list for each top-K retrieved image. We then choose the image from the top-K that requires the minimum amount of edits required based on the length of the output discrepancies list.

3.5. Text-Guided Image Editing with InstructPix2Pix

The final component of the ARES-GEN pipeline applies the generated editing instructions with InstructPix2Pix. This model operates as a conditional diffusion model that was fine-tuned to follow natural language editing instructions. Under the hood, the model extends the Stable Diffusion architecture by conditioning the denoising process on both the input image as well as the text instruction. We use the pretrained InstructPix2Pix model from the diffusers library, which is a high-level Python library developed by Hugging Face for working with diffusion models and was built on top of PyTorch to simplify loading, running, and modifying diffusion models. The hyperparameters that are used for the model initialization are:

1. num_inference_steps: this sets the number of denoising steps during the reverse diffusion process. In other words, it indicates how many iterations the model goes through to transform pure noise into the final image. For relatively quick inference time.

2. guidance_scale: this controls how strong the text prompt influences the image generation, sometimes also called the classifier-free guidance scale. Higher value means that the model will comply more to the text prompt which could reduce image diversity. A lower value gives the model more freedom, although it might ignore the prompt.

3. image_guidance_scale: this controls how much the model puts attention on the original input image as op-

posed to just following the text prompt. This hyperparameter is very specific to image-to-image editing models like InstructPix2Pix. A higher value means that the model preserves a lot of the input image's structure, while a lower value gives more freedom for the model to change the image based on the instruction.

3.6. Error Handling and Pipeline Robustness

Due to the reliance of the ARES-GEN pipeline on multiple external components, error handling mechanisms become essential for maintaining system reliability.

3.6.1 API Failure Management

The system relies on OpenAI's GPT-4 API for the discrepancy analysis step and could introduce a point of failure. Motivated by this, we implemented a retry mechanism, where we use an exponential backoff strategy for transient API failures with initial retry delays of 1 second, and then increasing to 2, 4, and 8 second for the next attempts. The system performs up to 3 retry attempts before continuing with a simplified fallback approach. The simplified fallback mechanism basically ensures that the pipeline still returns an output although the OpenAI API does not work. In this case, the system will just retrieve an image with the highest CLIP similarity score and returns it right away, bypassing the discrepancy analysis.

3.6.2 Image Processing Error Handling

The image processing steps can fail due to corrupted files or unsupported formats. For this, before further processing, all images are passed into a basic format validation with the Python's PIL library. If an image fails to load properly, it will be excluded from the candidate pool.

3.6.3 Model Inference Timeout Handling

The InstructPix2Pix editing mechanism can sometimes take an unreasonable amount of time to complete. To navigate this, we implement a simple timeout management, where the editing operations are subject to a 300-second timeout. If editing exceeds this threshold, the process will be terminated and the system will return the best unedited retrieved image with a notification that editing was unsuccessful. This helps with preventing users from indefinite waiting and ensures that the model still provides output to the user.

4. Dataset and Features

For this project, we used the Flickr30k dataset, which consists of approximately 31.8k images collected from Flickr [18]. This dataset is publicly available through Kaggle. The Flickr30k dataset contains various everyday scenes including including people, animals, outdoor activities, urban environments, and social gatherings. Each image in the original dataset comes with descriptive captions, though these are not used in our current implementation.

4.1. Dataset Preprocessing and Subset Creation

Due to computational constraints and to ensure reasonable processing time when running locally, we created a subset of 1000 images from the original Flickr30k dataset. The subset creation process involves the steps below:

1. Image filtering: only images with minimum dimensions of 224x224 pixels were included to ensure decent resolution

2. Format standardization: all images were converted to RGB before processing

3. Random sampling: images were randomly sampled from the original dataset to maintain the distribution across different scene types and content

4. File validation: each image was validated for proper format before inclusion.

4.2. Feature Extraction

We used the Hugging Face's CLIP model, in particular clip-vit-base-patch32 to embed the images within the subset. The technical implementations are as follows:

1. Model initialization: the pre-trained CLIP model was loaded

2. Batch processing: images were processed in batches of 32 to ensure computational efficiency

3. Preprocessing: each image was automatically resized and normalized according to CLIP's requirements using the CLIPProcessor class also coming from Hugging Face's transformers library.

4. Feature extraction: The vision encoder generated embedding vectors vectors for each image, and were L2-normalized to unit vectors to allow cosine similarity computations.

4.3. Data Storage

The embeddings for each image were then saved as a numpy array of embeddings in a .npy format. We also built a metadata in a form of .txt file containing corresponding image file paths for better traceability.

5. Experiments, Results, and Discussions

Our ARES-GEN pipeline utilizes pre-trained CLIP, BLIP, and InstructPix2Pix model.For the CLIP-based image retrieval, we use a batch size of 32 for embedding generation to ensure memory efficiency and training speed. We also retrieved top-K with $k = 5$ images based on the cosine similarity scores. For the BLIP captioning model, we used num_beams=3 and max_new_tokens=64 to generate concise descriptive captions. We also used length_penalty=1.1 to force slightly longer outputs and no_repeat_ngram_size=2 to prevent repetitive phrases.

For the InstructPix2Pix model, we use num_inference_steps=30 to ensure decent image quality while maintaining reasonable inference time, guidance_scale=7.5 to ensure compliance to the query, and image_guidance_scale=7.5 to ensure preserve structural elements of the original image while still allows a certain degree of modifications. No cross-validation was performed since all components use pre-trained models without fine-tuning.

5.1. Results

5.1.1 CLIP Similarity

For the first evaluation metric, we calculated the CLIP similarity pre (right after the retrieval) and post editing on 30 data points between the images and the user queries. The results are shown below:

Figure 2. CLIP Similarity Scores

We can see that 73.3% of the time, the CLIP similarity score increases after the editing process, which is not necessarily a good thing as it will be discussed on the qualitative result section.

Some of the prompts we used for the evaluation purposes are as follows: To give a clear depiction of what's happening, consider the second example where we have the prompt "A baby sitting on a green bean bag". Within the dataset, we have a semantically similar image but with a blue bean bag, as shown below: The critic agent then recognizes that there

Initial Prompt	Recommended Edit
A mom carrying a baby with a blue attire.	Change the little girl's attire to blue
Ababy sitting on a green bean bag.	Change the bean bag's color to green

Figure 3. Evaluation Prompts

Figure 4. Example 1 of a Retrieved Image

is a disparity in the color of the sofa, and thus the editing agent applied the edit and produced the following image:

Figure 5. Example 1 of a Post-Edit Image

5.1.2 Qualitative Analysis and Limitations

Although the quantitative results demonstrate a consistent improvement in CLIP similarity scores from the editing process, a qualitative analysis reveals several important limitations to the current implementation of ARES-GEN. Manual inspections have shown that post-edit images are not always qualitatively superior to their pre-edit counterparts.

One of the limitations observed is the model's difficulty in handling complex edits that require substantial updates. The InstructPix2Pix model is observed to be relatively good for simple attribute changes, such as color modifications as shown on figure 5, but struggles with more complex edits. One of the fail examples is when we gave an input

prompt of ”A woman sleeping with a dog besides her”. In the database, we have an image of a woman sleeping, which was retrieved correctly. However, when asked to add a dog, the results are not as expected, as shown below.

Figure 6. Example 2 of a Retrieved Image

Figure 7. Example 2 of a Post-Edit Image

The similarity scores of the two pictures above are more or less the same, but we could see a degradation of quality on figure 7.

Despite these challenges with complex edits, our system demonstrates a promising quality in preserving the integrity of the overall structure of the retrieved images. The editing maintains the spatial relationships between objects, lighting conditions, and the composition within the image. This preservation is key for maintaining coherence and ensures that the edited images have a photorealistic appearance, which was our main goal for this model.

However, we faced another significant technical limitation during the experiment, which is the degradation in image resolution that occurs post-edit. This can be seen clearly on figure 5 and figure 7. The InstructPix2Pix model seems to produce outputs at a lower resolution than the retrieved images. This struggle might stem from the diffusion model underlying the InstructPix2Pix model, which was trained on lower resolution images to reduce computational cost. Because of this, the iterative denoising process might introduce artifacts and blur fine details, especially when making localized edits that require accurate spatial control.

The model’s limited ability for deep semantic understanding could also contribute to its poor performance on complex edits. Unlike human who can easily learn the relationships between objects and maintain logical consistency during the edit, InstructPix2Pix operates at the pixel level with limited scene comprehension. Below is another example of failed attempt:

Figure 8. Retrieved Image, Prompt: ”A woman carrying a baby with a blue attire”

6. Conclusion and Future Work

In this work, we introduced a novel retrieval-and-edit pipeline for context-aware text-to-image generation that uses existing high-resolution images and applies edits to match the user query better. Our approach demonstrated

Figure 9. Edited Image: Wrong Target for the Edit

quantitative improvements in CLIP similarity scores and effectively applied simple attribute modifications while maintaining the structure of the overall image. However, qualitative evaluation showed limitations in handling complex edits and resolution degradation due to InstructPix2Pix constraints. The primary bottleneck comes from computational limitations, since all experiments were conducted on CPU hardware locally. This limits our ability to explore more sophisticated pre-trained models. For future work, access to GPU resources could enable exploration of more advanced diffusion-based editors and large-scale vision-language models. Additionally, integrating more complex agentic pipeline could also help with preventing hallucinations.

References

[1] Alex Clark and Contributors. Pillow (pil fork). `https://python-pillow.org`, 2024. Accessed: 2025-06-04.

[2] C. da Costa-Luis and contributors. tqdm: A fast, extensible progress bar for python and cli. `https://github.com/tqdm/tqdm`, 2024. Accessed: 2025-06-04.

[3] O. Gafni, A. Polyak, O. Ashual, S. Sheynin, D. Parikh, and Y. Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision (ECCV)*, pages 517–534. Springer Nature Switzerland, 2022.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[5] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

[6] Hugging Face. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022. Accessed: 2025-06-04.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment, 2007.

[8] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pages 4904–4916. PMLR, 2021.

[9] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, and I. Sutskever. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[10] X. Pan, X. Zuo, B. Dai, and C. C. Loy. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 Conference Proceedings*. ACM, 2023.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. `https://pytorch.org/`, 2019. Accessed: 2025-06-04.

[12] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[13] A. Ramesh, P. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1.2:3, 2022.

[14] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, T. Salimans, J. Ho, D. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494, 2022.

[15] A. Singh, D. Mahajan, M. Rohrbach, I. Misra, and P. Goyal. FLAVA: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15638–15650, 2022.

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing, 2020.

[17] B. Yang, X. Lin, Y. Ji, R. Zhang, H. Zhang, S. Bai, J. Kautz, T. Zhang, and W. Wang. Paint by example: Exemplar-based image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17188–17200, 2023.

[18] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

7. Libraries

For this project, we used numpy [5], PyTorch [11], pillow [1], diffusers [6], matplotlib [7], tqdm [2], and transformers [16].